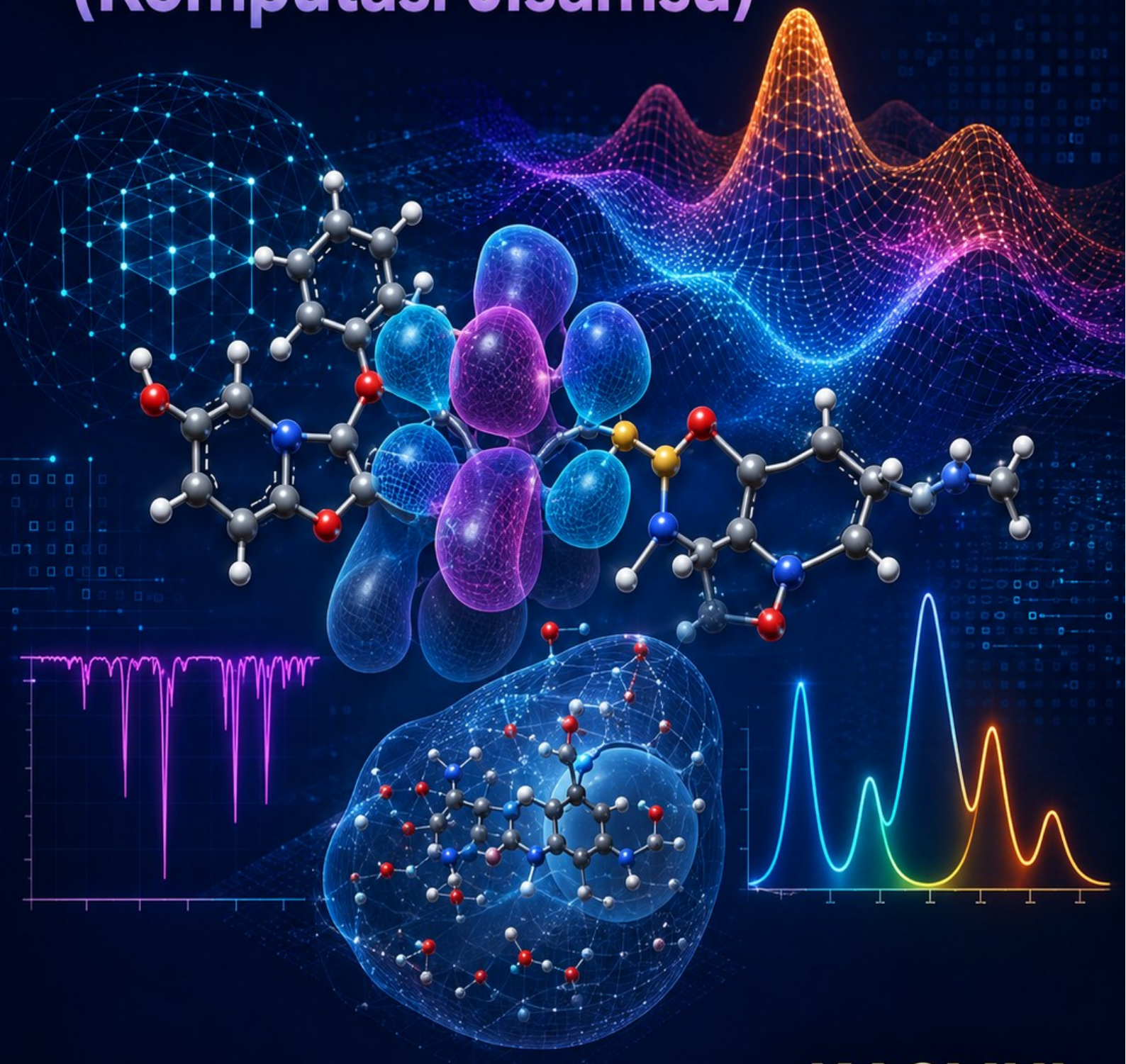
APLIKASI QM KOMPUTASI V2.3.4

(Komputasi Jisamsu)



https://kasmui.cloud/qmkomputasi/

KASMUI

BUKU REFERENSI DAN PANDUAN TEKNIS

APLIKASI
QM KOMPUTASI (QMCA) V2.3.4

Komputasi Jisamsu

Gambar 1. Peta menyeluruh komponen aplikasi yang didokumentasikan dalam buku.

PySCF • RDKit • MOPAC • xTB • ORCA

Edisi Dokumentasi V2.3.4 — 2026

INFORMASI BUKU

Judul: APLIKASI QM KOMPUTASI (QMCA) V2.3.4 (Komputasi Jisamsu). Buku ini disusun dari tutorial instalasi DOCX dan kode sumber Python v2.3.4 yang dilampirkan. Fitur, nama menu, metode, basis, job, add-on, struktur folder, dan perilaku aplikasi ditelusuri dari sumber tersebut; konsep teoretis didukung oleh literatur pada Daftar Pustaka.

Cakupan: instalasi WSL/Debian, input struktur, RDKit, PySCF, semiempirik, MOPAC, xTB, ORCA, HF/MP2/DFT, basis set, optimasi, frekuensi, IR, termokimia, UV–Vis, PCM, NMR, Raman, G3MP2, NBO, orbital, MEP, cube, QSAR/QSPR, manajemen job, troubleshooting, latihan, glosarium, referensi fungsi, dan lampiran kode sumber.

Website pendukung: https://kasmui.cloud/qmkomputasi/

Catatan penggunaan: Buku ini adalah panduan ilmiah dan teknis. Keberhasilan program mengeksekusi job tidak otomatis membuktikan bahwa model kimia, keadaan spin, atau interpretasi hasil telah benar.

KATA PENGANTAR

Komputasi kimia telah berkembang dari kumpulan perintah yang hanya nyaman bagi pengguna khusus menjadi ekosistem kerja yang dapat dijalankan melalui bahasa pemrograman, notebook, antarmuka grafis, dan layanan web lokal. Perubahan ini membuka peluang besar bagi mahasiswa dan peneliti untuk mempelajari struktur elektronik, optimasi geometri, spektrum, termokimia, serta deskriptor molekul secara lebih terintegrasi. Namun kemudahan antarmuka juga menghadirkan tanggung jawab baru: pengguna harus mampu membedakan antara pekerjaan yang berhasil dijalankan dan hasil yang sahih secara ilmiah.

Buku ***APLIKASI QM KOMPUTASI (QMCA) V2.3.4 (Komputasi Jisamsu)*** disusun untuk menjembatani dua kebutuhan tersebut. Di satu sisi, buku menjelaskan langkah operasional mulai dari pengunduhan paket, instalasi Debian pada WSL/Ubuntu, pengelolaan katalog molekul, pemilihan backend, hingga pembacaan folder hasil. Di sisi lain, buku memberi landasan untuk memahami mengapa charge, multiplicity, metode, basis set, toleransi SCF, grid DFT, optimasi, analisis frekuensi, model pelarut, dan prosedur validasi tidak boleh dipilih secara mekanis.

Sumber utama buku adalah berkas tutorial dan kode Python v2.3.4 yang dilampirkan. Kode tersebut memperlihatkan aplikasi Streamlit yang memadukan RDKit untuk identitas, konektivitas, visualisasi, dan geometri awal dengan beberapa jalur komputasi: PySCF, ekstensi semiempirik PySCF, MOPAC, xTB, dan ORCA. Aplikasi juga menyediakan add-on IR, Thermodynamics, UV–Vis, NMR, Raman, G3MP2, dan NBO; membuat file cube; mengekspor struktur; menghitung deskriptor; menjalankan job di thread latar; serta menyimpan konfigurasi dan hasil pada folder proyek.

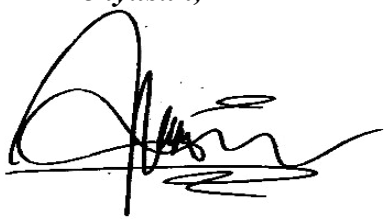
Penyusunan buku dilakukan dengan prinsip keterlacakan. Nama fitur dan perilaku aplikasi diturunkan dari kode sumber. Ketika terdapat keterbatasan atau risiko implementasi, buku menyatakannya secara terbuka. Salah satu contohnya ialah penggunaan APP_DIR sebagai lokasi qm_runs dan upload_molekul.json pada sumber terlampir. Pada instalasi di direktori sistem seperti /opt, pendekatan ini harus dipastikan memperoleh lokasi tulis yang aman melalui rancangan paket atau perubahan kode; menjalankan aplikasi dengan hak root bukan solusi yang disarankan untuk penggunaan rutin.

Buku ini dirancang untuk beberapa kelompok pembaca. Mahasiswa dapat menggunakannya sebagai panduan praktikum dan sumber latihan. Dosen dapat mengadaptasi studi kasus, pertanyaan, serta rubrik validasi untuk pembelajaran komputasi kimia. Peneliti dapat memakai bagian workflow, reproduksibilitas, dan pemecahan masalah sebagai daftar periksa ketika menyusun protokol. Pengembang perangkat lunak dapat memanfaatkan referensi fungsi dan lampiran kode untuk menelusuri arsitektur aplikasi serta merancang pengujian regresi.

Ucapan terima kasih ditujukan kepada komunitas pengembang perangkat lunak ilmiah terbuka yang memungkinkan workflow seperti ini dibangun: PySCF, Psi4 sebagai rujukan ekosistem komputasi kuantum terbuka, RDKit, Streamlit, SciPy, NumPy, pandas, geomeTRIC, xTB, dan berbagai proyek lain. Perangkat lunak eksternal dengan lisensi khusus tetap harus digunakan sesuai ketentuan masing-masing. Semoga buku ini membantu pembaca menjalankan komputasi secara lebih tertib, kritis, dan bertanggung jawab—serta menjadikan setiap hasil bukan hanya mudah diperoleh, tetapi juga dapat diuji dan dipertanggungjawabkan.

Semarang, Agustus 2026

Penyusun,



Kasmui

PETUNJUK MENGGUNAKAN BUKU

Buku dapat dibaca secara berurutan atau digunakan sebagai referensi. Bab 1–4 membangun orientasi aplikasi dan lingkungan; Bab 5–8 membahas metode dan properti; Bab 9 mengajarkan interpretasi keluaran; Bab 10 menyatukan workflow, troubleshooting, dan latihan. Glosarium membantu pencarian istilah, sedangkan lampiran fungsi dan kode memudahkan audit teknis.

1. Mulai dari Bab 1 bila belum terbiasa dengan hubungan struktur, metode, basis, dan properti.
2. Gunakan Bab 2 sebagai checklist instalasi; jangan melanjutkan sebelum smoke test berhasil.
3. Sebelum setiap job, tulis tujuan, charge, multiplicity, metode, basis, dan kriteria validasi.
4. Setelah job, periksa log, struktur akhir, konvergensi, frekuensi, satuan, dan kelengkapan file.
5. Tekan Ctrl+A lalu F9 di Microsoft Word untuk memperbarui daftar isi dan semua nomor halaman.

DAFTAR ISI

BAB 1 — ORIENTASI APLIKASI DAN LANDASAN KOMPUTASI MOLEKUL

Bab ini menempatkan aplikasi sebagai jembatan antara struktur kimia, model mekanika kuantum, dan keputusan penelitian. Fokusnya bukan hanya cara menekan tombol, melainkan cara membangun pertanyaan, memilih tingkat teori, dan menilai apakah keluaran layak dipercaya.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Ruang Lingkup dan Tujuan Aplikasi

Ruang Lingkup dan Tujuan Aplikasi merupakan bagian penting dari alur komputasi karena aplikasi menyatukan persiapan molekul, pemilihan backend, eksekusi job, dan pembacaan hasil dalam satu alur yang dapat diaudit. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan ruang lingkup dan tujuan aplikasi sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada judul aplikasi, keterangan fitur, sidebar input, pengaturan komputasi, monitor job, dan panel hasil. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah tentukan sejak awal apakah tujuan utama adalah penyaringan, optimasi struktur, spektrum, termokimia, atau sifat elektronik. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah menggunakan semua keluaran sekaligus tanpa pertanyaan penelitian yang jelas sehingga interpretasi menjadi dangkal. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menyusun matriks tujuan–metode–keluaran dan memastikan setiap kolom memiliki alasan ilmiah. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Mahasiswa membandingkan benzena, toluena, dan fenol untuk mempelajari pengaruh substituen terhadap energi orbital dan momen dipol. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk ruang lingkup dan tujuan aplikasi dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa judul aplikasi, keterangan fitur, sidebar input, pengaturan komputasi, monitor job, dan panel hasil.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menggunakan semua keluaran sekaligus tanpa pertanyaan penelitian yang jelas sehingga interpretasi menjadi dangkal.

4. Terapkan validasi: menyusun matriks tujuan–metode–keluaran dan memastikan setiap kolom memiliki alasan ilmiah.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** tentukan sejak awal apakah tujuan utama adalah penyaringan, optimasi struktur, spektrum, termokimia, atau sifat elektronik.

Dari Struktur Kimia ke Hamiltonian

Pembahasan dari struktur kimia ke hamiltonian dimulai dari tujuan ilmiahnya: komputasi elektronik mengubah daftar atom, koordinat, muatan, dan keadaan spin menjadi persoalan energi yang diselesaikan secara hampiran. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan fungsi konversi molekul RDKit ke atom PySCF, penetapan charge dan multiplicity, serta pemilihan backend. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, periksa jumlah elektron dan konsistensi spin sebelum memilih RHF, ROHF, UHF, atau DFT. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena muatan atau multiplicity tidak cocok dengan jumlah elektron sehingga SCF gagal atau memberi keadaan elektronik yang salah. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan menghitung paritas elektron, memeriksa pesan backend, dan membandingkan beberapa tebakan awal untuk sistem terbuka. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Radikal organik netral tidak boleh dipaksa menjadi singlet hanya karena nilai default multiplicity adalah 1. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk dari struktur kimia ke hamiltonian dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa fungsi konversi molekul RDKit ke atom PySCF, penetapan charge dan multiplicity, serta pemilihan backend.
3. Jalankan uji kecil, kemudian periksa risiko berikut: muatan atau multiplicity tidak cocok dengan jumlah elektron sehingga SCF gagal atau memberi keadaan elektronik yang salah.
4. Terapkan validasi: menghitung paritas elektron, memeriksa pesan backend, dan membandingkan beberapa tebakan awal untuk sistem terbuka.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** periksa jumlah elektron dan konsistensi spin sebelum memilih RHF, ROHF, UHF, atau DFT.

Energi, Struktur, dan Properti

Energi, Struktur, dan Properti merupakan bagian penting dari alur komputasi karena energi total bukan nilai absolut yang berdiri sendiri; maknanya muncul melalui perbandingan yang konsisten dan turunan energi terhadap koordinat atau medan. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan energi, struktur, dan properti sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada konversi Hartree ke eV, kJ mol^{-1}, kcal mol^{-1}, dan cm^{-1} serta pembuatan tabel geometri dan orbital. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah bandingkan hanya energi dari jumlah elektron, Hamiltonian, basis, dan kondisi yang kompatibel. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah menafsirkan molekul dengan komposisi berbeda hanya dari energi total paling negatif. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menggunakan energi relatif, reaksi isodesmik bila perlu, dan membedakan energi elektronik dari entalpi atau energi bebas. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Dua konformer dibandingkan dengan ΔE dan, bila frekuensi tersedia, dengan ΔG pada temperatur yang sama. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk energi, struktur, dan properti dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa konversi Hartree ke eV, kJ mol^{-1}, kcal mol^{-1}, dan cm^{-1} serta pembuatan tabel geometri dan orbital.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menafsirkan molekul dengan komposisi berbeda hanya dari energi total paling negatif.
4. Terapkan validasi: menggunakan energi relatif, reaksi isodesmik bila perlu, dan membedakan energi elektronik dari entalpi atau energi bebas.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** bandingkan hanya energi dari jumlah elektron, Hamiltonian, basis, dan kondisi yang kompatibel.

Hierarki Model Komputasi

Pembahasan hierarki model komputasi dimulai dari tujuan ilmiahnya: tingkat teori membentuk kompromi antara biaya, korelasi elektron, ketahanan numerik, dan jenis sifat yang dapat dihitung. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan pilihan force field awal, semiempirik, HF, MP2, DFT, serta add-on komposit dan analisis khusus. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan metode murah untuk penyaringan dan metode lebih kuat untuk titik keputusan yang sensitif. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena menganggap metode yang lebih mahal selalu lebih akurat untuk semua jenis ikatan dan sifat. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan melakukan benchmark kecil pada sistem representatif dan membandingkan kecenderungan, bukan satu angka saja. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Seratus kandidat disaring dengan GFN2-xTB, lalu sepuluh terbaik dihitung ulang dengan DFT. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk hierarki model komputasi dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa pilihan force field awal, semiempirik, HF, MP2, DFT, serta add-on komposit dan analisis khusus.

3. Jalankan uji kecil, kemudian periksa risiko berikut: menganggap metode yang lebih mahal selalu lebih akurat untuk semua jenis ikatan dan sifat.
4. Terapkan validasi: melakukan benchmark kecil pada sistem representatif dan membandingkan kecenderungan, bukan satu angka saja.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: gunakan metode murah untuk penyaringan dan metode lebih kuat untuk titik keputusan yang sensitif.

Reproduksibilitas dan Jejak Audit

Reproduksibilitas dan Jejak Audit merupakan bagian penting dari alur komputasi karena hasil ilmiah harus dapat ditelusuri dari identitas molekul, versi program, konfigurasi, log, sampai file keluaran. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan reproduksibilitas dan jejak audit sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada folder run unik, config.json, input_manifest.json, log progres, hasil per molekul, dan ekspor struktur. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah arsipkan folder job secara utuh dan jangan memisahkan tabel hasil dari konfigurasi sumbernya. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah menyalin angka ke spreadsheet tanpa mencatat metode, basis, seed, atau struktur akhir. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah mengulang satu job dari konfigurasi tersimpan dan memeriksa kesetaraan struktur serta energi dalam toleransi yang wajar. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Satu hasil publikasi direkonstruksi enam bulan kemudian dari folder proyek tanpa mengandalkan ingatan pengguna. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk reproduksibilitas dan jejak audit dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa folder run unik, config.json, input_manifest.json, log progres, hasil per molekul, dan ekspor struktur.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menyalin angka ke spreadsheet tanpa mencatat metode, basis, seed, atau struktur akhir.
4. Terapkan validasi: mengulang satu job dari konfigurasi tersimpan dan memeriksa kesetaraan struktur serta energi dalam toleransi yang wajar.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: arsipkan folder job secara utuh dan jangan memisahkan tabel hasil dari konfigurasi sumbernya.

Batas Aplikasi dan Tanggung Jawab Pengguna

Pembahasan batas aplikasi dan tanggung jawab pengguna dimulai dari tujuan ilmiahnya: antarmuka memudahkan operasi tetapi tidak menggantikan diagnosis keadaan elektronik, kualitas geometri, atau kecukupan model. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan pesan peringatan backend, error input, frekuensi imajiner, dan keterangan bahwa RDKit memberi geometri awal sedangkan backend menghitung QM. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, perlakukan setiap peringatan sebagai item investigasi, bukan gangguan tampilan. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena menganggap status selesai identik dengan hasil benar secara kimia. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

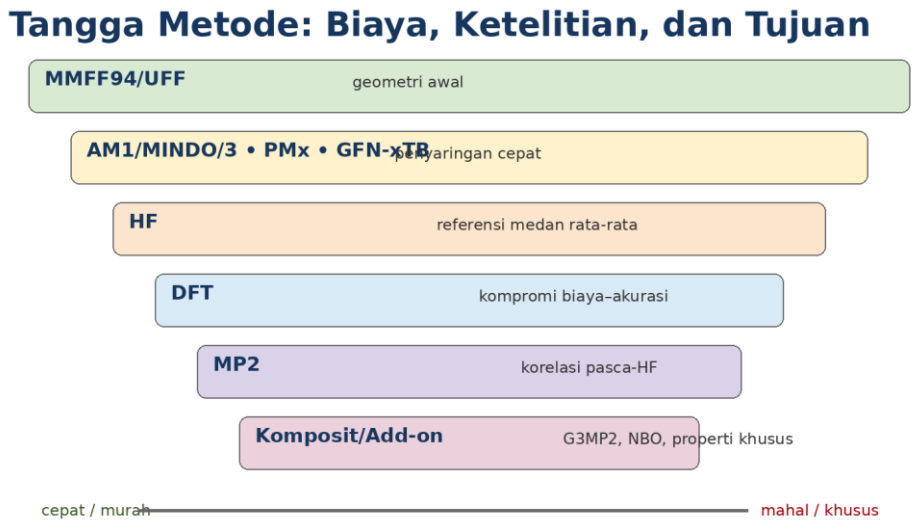
Bukti bahwa prosedur layak dipakai diperoleh dengan membuat checklist kelulusan yang mencakup konvergensi, struktur, frekuensi, satuan, dan sensitivitas model. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Optimasi selesai tetapi satu ikatan terputus; hasil harus ditolak meskipun log tidak berakhir dengan exception. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk batas aplikasi dan tanggung jawab pengguna dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa pesan peringatan backend, error input, frekuensi imajiner, dan keterangan bahwa RDKit memberi geometri awal sedangkan backend menghitung QM.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menganggap status selesai identik dengan hasil benar secara kimia.
4. Terapkan validasi: membuat checklist kelulusan yang mencakup konvergensi, struktur, frekuensi, satuan, dan sensitivitas model.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: perlakukan setiap peringatan sebagai item investigasi, bukan gangguan tampilan.



Gambar 2. Tangga metode dari persiapan geometri hingga metode elektronik dan analisis khusus.

Tabel 1. Hubungan tujuan, job, dan validasi utama

Tujuan	Job/fitur	Validasi
Energi elektronik	Single Point	SCF, keadaan spin, basis, satuan
Struktur minimum	Geometry Optimization	Gradien dan frekuensi
Spektrum IR	Frequency / Add-on IR	Mode normal, faktor skala
Termokimia	Thermodynamics	T, P, ZPE, mode rendah
UV–Vis	Add-on UV-Vis	Akar, oscillator strength, karakter transisi

BAB 2 — INSTALASI, PAKET DEBIAN, WSL, DAN LINGKUNGAN KERJA

Bab ini mengembangkan tutorial terlampir menjadi prosedur instalasi, verifikasi, pemutakhiran, dan diagnosis yang dapat digunakan pada Windows dengan WSL/Ubuntu. Perhatian khusus diberikan pada hak akses direktori, executable eksternal, dan keterulangan lingkungan.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Persiapan Windows dan WSL

Pembahasan persiapan windows dan wsl dimulai dari tujuan ilmiahnya: WSL menyediakan lingkungan Linux yang terintegrasi dengan Windows tetapi memiliki sistem path, izin, dan jaringan yang perlu dipahami. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan akses folder melalui Explorer, perintah wsl, localhost Streamlit, dan Network URL yang ditampilkan saat aplikasi berjalan. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, simpan paket instalasi di lokasi yang mudah dicapai dari WSL dan gunakan path Linux ketika menjalankan perintah. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena mencampur path C:\ dengan /mnt/c atau menjalankan paket dari shell yang tidak tepat. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan memastikan `wsl --status`, distribusi Ubuntu, Python, dan akses folder kerja berfungsi sebelum instalasi. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Berkas ZIP di Desktop Windows diekstrak lalu dibuka dari terminal WSL tanpa menyalin ulang ke lokasi lain. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk persiapan windows dan wsl dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa akses folder melalui Explorer, perintah wsl, localhost Streamlit, dan Network URL yang ditampilkan saat aplikasi berjalan.
3. Jalankan uji kecil, kemudian periksa risiko berikut: mencampur path C:\ dengan /mnt/c atau menjalankan paket dari shell yang tidak tepat.
4. Terapkan validasi: memastikan `wsl --status`, distribusi Ubuntu, Python, dan akses folder kerja berfungsi sebelum instalasi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** simpan paket instalasi di lokasi yang mudah dicapai dari WSL dan gunakan path Linux ketika menjalankan perintah.

Instalasi Paket .deb

Instalasi Paket .deb merupakan bagian penting dari alur komputasi karena paket Debian memasang file aplikasi, launcher, dan metadata paket melalui apt/dpkg sehingga dapat dikelola oleh sistem. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan instalasi paket .deb sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada perintah `sudo apt install ./app_qm_streamlit_v2.3.4.deb`, perbaikan dependensi, dan pemeriksaan `dpkg -l`. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah gunakan apt dengan awalan `./` untuk file lokal dan baca nama paket aktual dari metadata. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah nama file .deb tidak sama dengan nama paket sehingga perintah uninstall atau grep keliru. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah memeriksa status ii pada dpkg, lokasi launcher dengan `command -v`, dan versi yang dilaporkan aplikasi. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Paket terpasang tetapi launcher tidak ditemukan karena terminal lama belum menyegarkan PATH. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk instalasi paket .deb dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa perintah `sudo apt install ./app_qm_streamlit_v2.3.4.deb`, perbaikan dependensi, dan pemeriksaan `dpkg -l`.
3. Jalankan uji kecil, kemudian periksa risiko berikut: nama file .deb tidak sama dengan nama paket sehingga perintah uninstall atau grep keliru.
4. Terapkan validasi: memeriksa status ii pada dpkg, lokasi launcher dengan `command -v`, dan versi yang dilaporkan aplikasi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan apt dengan awalan `./` untuk file lokal dan baca nama paket aktual dari metadata.

Dependensi Python dan Executable Eksternal

Pembahasan dependensi python dan executable eksternal dimulai dari tujuan ilmiahnya: dependensi pip dapat diimpor atau dipasang otomatis, sedangkan ORCA, MOPAC, xTB, Open Babel, Gaussian, dan NBO memerlukan instalasi sistem atau lisensi terpisah. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan fungsi ensure_python_package dan backend_status yang membedakan modul Python dari executable pada PATH. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, kelola versi Python dalam lingkungan yang konsisten dan pasang executable eksternal sesuai lisensi resmi. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena auto-install ke interpreter yang tidak sama dengan launcher sehingga modul tetap tidak ditemukan. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan menjalankan `python -c` pada interpreter aktif, `which` untuk executable, dan uji job sangat kecil per backend. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: PySCF berhasil tetapi add-on Raman gagal karena ORCA tidak tersedia pada PATH. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk dependensi python dan executable eksternal dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa fungsi ensure_python_package dan backend_status yang membedakan modul Python dari executable pada PATH.
3. Jalankan uji kecil, kemudian periksa risiko berikut: auto-install ke interpreter yang tidak sama dengan launcher sehingga modul tetap tidak ditemukan.
4. Terapkan validasi: menjalankan `python -c` pada interpreter aktif, `which` untuk executable, dan uji job sangat kecil per backend.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: kelola versi Python dalam lingkungan yang konsisten dan pasang executable eksternal sesuai lisensi resmi.

Direktori Data dan PermissionError

Direktori Data dan PermissionError merupakan bagian penting dari alur komputasi karena aplikasi terpasang di /opt biasanya tidak dapat menulis ke direktori program bagi pengguna biasa. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan direktori data dan permissionerror sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada kode sumber membentuk RUNS_DIR = APP_DIR / 'qm_runs' dan UPLOAD_MOLECULES_FILE di APP_DIR. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah paket produksi perlu mengarahkan data berubah ke direktori pengguna seperti ~/.local/share/app-qm-streamlit-v2.3.4. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah PermissionError terjadi saat import sebelum antarmuka tampil atau saat menyimpan katalog molekul. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menjalankan launcher tanpa sudo, membuat satu job, menambah satu molekul, lalu memastikan berkas tersimpan di lokasi yang dapat ditulis. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Instalasi di /opt berhasil, tetapi aplikasi gagal pada baris RUNS_DIR.mkdir karena pengguna tidak memiliki izin tulis. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk direktori data dan permissionerror dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa kode sumber membentuk RUNS_DIR = APP_DIR / 'qm_runs' dan UPLOAD_MOLECULES_FILE di APP_DIR.
3. Jalankan uji kecil, kemudian periksa risiko berikut: PermissionError terjadi saat import sebelum antarmuka tampil atau saat menyimpan katalog molekul.
4. Terapkan validasi: menjalankan launcher tanpa sudo, membuat satu job, menambah satu molekul, lalu memastikan berkas tersimpan di lokasi yang dapat ditulis.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** paket produksi perlu mengarahkan data berubah ke direktori pengguna seperti ~/.local/share/app-qm-streamlit-v2.3.4.

Menjalankan dan Mengakses Streamlit

Pembahasan menjalankan dan mengakses streamlit dimulai dari tujuan ilmiahnya: server Streamlit berjalan pada port tertentu dan browser Windows dapat mengakses localhost WSL melalui integrasi jaringan. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan output Local URL http://localhost:8501, Network URL, dan pesan file watcher untuk WSL. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan localhost untuk komputer yang sama dan Network URL hanya bila jaringan serta firewall mengizinkan. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena menganggap Network URL stabil padahal alamat WSL dapat berubah setelah restart. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan menguji kedua URL, memeriksa port, dan memastikan proses launcher tetap hidup di terminal. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Halaman browser tidak terbuka karena proses dihentikan saat terminal ditutup. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk menjalankan dan mengakses streamlit dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa output Local URL http://localhost:8501, Network URL, dan pesan file watcher untuk WSL.

3. Jalankan uji kecil, kemudian periksa risiko berikut: menganggap Network URL stabil padahal alamat WSL dapat berubah setelah restart.
4. Terapkan validasi: menguji kedua URL, memeriksa port, dan memastikan proses launcher tetap hidup di terminal.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: gunakan localhost untuk komputer yang sama dan Network URL hanya bila jaringan serta firewall mengizinkan.

Uninstall, Upgrade, dan Pemulihan

Uninstall, Upgrade, dan Pemulihan merupakan bagian penting dari alur komputasi karena siklus hidup paket mencakup pencadangan data pengguna, penghapusan paket, pembersihan konfigurasi, dan instalasi versi baru. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan uninstall, upgrade, dan pemulihan sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada identitas paket dari dpkg dan katalog upload_molekul.json yang perlu dipertahankan. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah cadangkan katalog serta folder run sebelum purge dan jangan menghapus data penelitian secara otomatis. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah upgrade menimpa file JSON atau launcher menunjuk versi lama. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah membandingkan versi sebelum–sesudah, menjalankan smoke test, dan memulihkan katalog dengan skema yang kompatibel. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Pengguna memperbarui v2.3.4 tanpa kehilangan molekul tersimpan dan hasil job sebelumnya. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk uninstall, upgrade, dan pemulihan dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa identitas paket dari dpkg dan katalog upload_molekul.json yang perlu dipertahankan.
3. Jalankan uji kecil, kemudian periksa risiko berikut: upgrade menimpa file JSON atau launcher menunjuk versi lama.
4. Terapkan validasi: membandingkan versi sebelum–sesudah, menjalankan smoke test, dan memulihkan katalog dengan skema yang kompatibel.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: cadangkan katalog serta folder run sebelum purge dan jangan menghapus data penelitian secara otomatis.

Gambar 3. Tampilan awal QM Komputasi V2.3.4 dari tutorial sumber.

Tabel 2. Perintah instalasi dan fungsi pemeriksaan

Perintah	Fungsi
wsl	Masuk ke distribusi WSL dari terminal Windows
sudo apt install ./app_qm_streamlit_v2.3.4.deb	Memasang paket lokal
sudo apt --fix-broken install	Memperbaiki dependensi paket
dpkg -l | grep app-qm-streamlit	Memeriksa status paket
app-qm-streamlit-v2.3.4	Menjalankan aplikasi

BAB 3 — INPUT MOLEKUL, IDENTITAS KIMIA, DAN RDKIT

Bab ini membahas seluruh jalur input yang diimplementasikan kode: preset, SMILES, InChI, path lokal, dan upload. RDKit digunakan untuk identitas, konektivitas, visualisasi, dan geometri awal; peran ini harus dibedakan dari perhitungan energi kuantum.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Lima Jalur Input Molekul

Lima Jalur Input Molekul merupakan bagian penting dari alur komputasi karena setiap jalur input membawa informasi dan risiko berbeda, dari string konektivitas hingga koordinat tiga dimensi lengkap. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan lima jalur input molekul sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada multiselect preset, kotak SMILES, kotak InChI, path lokal, dan file_uploader. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah pilih format paling kaya yang tersedia dan simpan sumber asli untuk audit. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah kehilangan stereokimia, protonasi, atau koordinat karena konversi format yang tidak tepat. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah membandingkan formula, InChIKey, muatan, stereokimia, dan gambar 2D/3D setelah impor. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Ligand kiral diimpor dari SDF dan dibandingkan dengan SMILES untuk memastikan pusat stereogenik tetap sama. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk lima jalur input molekul dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa multiselect preset, kotak SMILES, kotak InChI, path lokal, dan file_uploader.
3. Jalankan uji kecil, kemudian periksa risiko berikut: kehilangan stereokimia, protonasi, atau koordinat karena konversi format yang tidak tepat.
4. Terapkan validasi: membandingkan formula, InChIKey, muatan, stereokimia, dan gambar 2D/3D setelah impor.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: pilih format paling kaya yang tersedia dan simpan sumber asli untuk audit.

SMILES Bernama dan Parsing

Pembahasan smiles bernama dan parsing dimulai dari tujuan ilmiahnya: format Nama=SMILES memisahkan label proyek dari representasi struktur tanpa memakai pemisah yang dapat muncul pada sintaks lain. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan _split_named_smiles, parse_smiles_lines, pembuatan nama unik, dan penyimpanan persisten. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan satu molekul per baris dan nama yang stabil, singkat, serta informatif. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena SMILES valensi tidak sah atau tanda sama dengan pada posisi yang membingungkan parser. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan memeriksa pesan error per baris dan melakukan round-trip SMILES kanonik. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Daftar analog 8a, 8c, 8s, dan 8w dimasukkan sebagai empat baris bernama. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk smiles bernama dan parsing dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa _split_named_smiles, parse_smiles_lines, pembuatan nama unik, dan penyimpanan persisten.
3. Jalankan uji kecil, kemudian periksa risiko berikut: SMILES valensi tidak sah atau tanda sama dengan pada posisi yang membingungkan parser.
4. Terapkan validasi: memeriksa pesan error per baris dan melakukan round-trip SMILES kanonik.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan satu molekul per baris dan nama yang stabil, singkat, serta informatif.

InChI, InChIKey, dan Identitas

InChI, InChIKey, dan Identitas merupakan bagian penting dari alur komputasi karena InChI merepresentasikan struktur berlapis, sedangkan InChIKey adalah pengenal ringkas untuk pencarian dan deduplikasi. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan inchi, inchikey, dan identitas sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada parse_inchi_lines dan molecule_identifiers yang menghasilkan SMILES, InChI, InChIKey, serta formula. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah simpan InChI penuh bersama InChIKey; jangan memakai InChIKey sebagai pengganti struktur. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah tautomer, protonasi, atau isotop berbeda disamakan secara tidak sengaja. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah membandingkan layer InChI, formula, muatan, dan struktur visual. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Dua bentuk protonasi senyawa basa diberi nama berbeda dan tidak digabungkan hanya karena nama trivial sama. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk inchi, inchikey, dan identitas dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa parse_inchi_lines dan molecule_identifiers yang menghasilkan SMILES, InChI, InChIKey, serta formula.
3. Jalankan uji kecil, kemudian periksa risiko berikut: tautomer, protonasi, atau isotop berbeda disamakan secara tidak sengaja.
4. Terapkan validasi: membandingkan layer InChI, formula, muatan, dan struktur visual.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: simpan InChI penuh bersama InChIKey; jangan memakai InChIKey sebagai pengganti struktur.

File Struktur dan Open Babel

Pembahasan file struktur dan open babel dimulai dari tujuan ilmiahnya: SDF, MOL, MOL2, PDB, XYZ, SMI, dan TXT memiliki tingkat informasi konektivitas yang berbeda. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan molecule_from_bytes, molecule_from_local_path, infer_bonds_xyz, dan fallback Open Babel. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan MOL/SDF untuk mempertahankan bond order, XYZ untuk koordinat saja, dan dokumentasikan konversi. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena bond inference keliru pada kompleks logam atau jarak antarmolekul dekat. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan memeriksa jumlah atom, ikatan, valensi, dan fragmentasi setelah impor. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: XYZ hasil optimasi eksternal dibaca dan konektivitasnya diperiksa sebelum dipakai menghitung deskriptor. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk file struktur dan open babel dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa molecule_from_bytes, molecule_from_local_path, infer_bonds_xyz, dan fallback Open Babel.
3. Jalankan uji kecil, kemudian periksa risiko berikut: bond inference keliru pada kompleks logam atau jarak antarmolekul dekat.
4. Terapkan validasi: memeriksa jumlah atom, ikatan, valensi, dan fragmentasi setelah impor.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan MOL/SDF untuk mempertahankan bond order, XYZ untuk koordinat saja, dan dokumentasikan konversi.

Geometri Awal, Hidrogen, dan Seed

Geometri Awal, Hidrogen, dan Seed merupakan bagian penting dari alur komputasi karena SMILES dua dimensi memerlukan penambahan hidrogen, embedding 3D, serta pre-optimasi sebelum menjadi input yang masuk akal. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan geometri awal, hidrogen, dan seed sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada build_rdkit_3d, prepare_molecule_3d, seed 20260820, dan opsi MMFF94/UFF. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah gunakan seed tetap untuk reproduksibilitas dan buat beberapa konformer untuk molekul fleksibel. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah satu konformer acak dianggap mewakili minimum global. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah mengulang embedding, membandingkan energi force field, dan melakukan optimasi QM pada kandidat terpilih. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Ibuprofen fleksibel memerlukan lebih dari satu konformer awal sebelum perbandingan sifat. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk geometri awal, hidrogen, dan seed dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa build_rdkit_3d, prepare_molecule_3d, seed 20260820, dan opsi MMFF94/UFF.
3. Jalankan uji kecil, kemudian periksa risiko berikut: satu konformer acak dianggap mewakili minimum global.
4. Terapkan validasi: mengulang embedding, membandingkan energi force field, dan melakukan optimasi QM pada kandidat terpilih.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: gunakan seed tetap untuk reproduksibilitas dan buat beberapa konformer untuk molekul fleksibel.

Katalog Persisten upload_molekul.json

Pembahasan katalog persisten upload_molekul.json dimulai dari tujuan ilmiahnya: input yang baru ditambahkan disimpan agar tersedia lagi setelah aplikasi restart dan diurutkan dengan tambahan terbaru di atas. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan read_json_safe, write_json_atomic, persist_molecule_spec, digest, dan build_molecule_catalog. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, perlakukan JSON sebagai data pengguna yang perlu dicadangkan dan divalidasi. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena penulisan di direktori tidak dapat ditulis, entri duplikat, atau MolBlock tidak dapat dibaca kembali. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan restart aplikasi, pilih molekul tersimpan, dan cocokkan fingerprint serta identitasnya. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Molekul hasil upload SDF muncul kembali pada multiselect setelah server dimulai ulang. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk katalog persisten upload_molekul.json dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa read_json_safe, write_json_atomic, persist_molecule_spec, digest, dan build_molecule_catalog.
3. Jalankan uji kecil, kemudian periksa risiko berikut: penulisan di direktori tidak dapat ditulis, entri duplikat, atau MolBlock tidak dapat dibaca kembali.
4. Terapkan validasi: restart aplikasi, pilih molekul tersimpan, dan cocokkan fingerprint serta identitasnya.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: perlakukan JSON sebagai data pengguna yang perlu dicadangkan dan divalidasi.

Tabel 3. Perbandingan format input molekul

Format	Kekuatan	Risiko
SMILES	Ringkas, mudah diketik	Koordinat 3D tidak tersedia
InChI	Identitas berlapis dan standar	Tidak praktis untuk penyuntingan manual
SDF/MOL	Konektivitas dan koordinat	Variasi dukungan properti
PDB	Kompatibel dengan visualisasi	Bond order sering terbatas
XYZ	Koordinat sederhana	Tidak menyimpan konektivitas

BAB 4 — ARSITEKTUR KODE, JOB LATAR, DAN MANAJEMEN DATA

Bab ini membaca aplikasi sebagai sistem perangkat lunak: bootstrap, status backend, session state, worker thread, stop event, log, manifest, hasil, serta mekanisme ekspor. Tujuannya agar pengguna tingkat lanjut mampu mendiagnosis masalah dan mengembangkan aplikasi dengan aman.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Bootstrap dan Import Terkendali

Pembahasan bootstrap dan import terkendali dimulai dari tujuan ilmiahnya: aplikasi mencoba mengimpor dependensi inti dan memasang paket Python yang hilang menggunakan interpreter aktif. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan ensure_python_package sebelum import NumPy, pandas, Streamlit, RDKit, dan PySCF. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan bootstrap hanya untuk paket yang dipetakan internal dan catat versi yang benar-benar terpasang. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena instalasi jaringan gagal, wheel tidak cocok, atau lingkungan produksi berubah tanpa review. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan uji pada lingkungan bersih, matikan jaringan untuk menguji pesan gagal, dan bekukan versi produksi. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Server baru tidak memiliki RDKit; bootstrap mencoba wheel biner dan memberikan error eksplisit bila gagal. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk bootstrap dan import terkendali dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa ensure_python_package sebelum import NumPy, pandas, Streamlit, RDKit, dan PySCF.
3. Jalankan uji kecil, kemudian periksa risiko berikut: instalasi jaringan gagal, wheel tidak cocok, atau lingkungan produksi berubah tanpa review.
4. Terapkan validasi: uji pada lingkungan bersih, matikan jaringan untuk menguji pesan gagal, dan bekukan versi produksi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan bootstrap hanya untuk paket yang dipetakan internal dan catat versi yang benar-benar terpasang.

Deteksi Backend dan Sumber Daya

Deteksi Backend dan Sumber Daya merupakan bagian penting dari alur komputasi karena ketersediaan modul atau executable harus diketahui sebelum job dibuat agar kegagalan cepat dan informatif. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan deteksi backend dan sumber daya sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada backend_status, find_mopac_executable, detect_gpu_info, dan status sidebar. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah jalankan smoke test per backend, bukan hanya memeriksa nama executable. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah executable ada tetapi lisensi, library dinamis, atau versi input tidak kompatibel. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menghitung molekul dua atom dan memeriksa keluaran parser. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: ORCA ditemukan di PATH tetapi tidak dapat dijalankan karena library runtime belum tersedia. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk deteksi backend dan sumber daya dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa backend_status, find_mopac_executable, detect_gpu_info, dan status sidebar.
3. Jalankan uji kecil, kemudian periksa risiko berikut: executable ada tetapi lisensi, library dinamis, atau versi input tidak kompatibel.
4. Terapkan validasi: menghitung molekul dua atom dan memeriksa keluaran parser.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** jalankan smoke test per backend, bukan hanya memeriksa nama executable.

Session State dan Snapshot Konfigurasi

Pembahasan session state dan snapshot konfigurasi dimulai dari tujuan ilmiahnya: rerun Streamlit memerlukan state eksplisit agar thread, folder, dan status tidak hilang pada setiap interaksi. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan init_state, is_job_running, config dictionary, current_run_dir, dan last_run_dir. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, anggap config.json sebagai kontrak job yang tidak boleh berubah selama eksekusi. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena widget baru tidak dimasukkan ke snapshot sehingga hasil tidak mencerminkan tampilan. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan bandingkan semua kunci UI dengan config.json dan uji perubahan saat job berjalan. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Pengguna mengubah basis setelah menekan Run; job aktif tetap memakai basis dalam snapshot awal. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk session state dan snapshot konfigurasi dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa init_state, is_job_running, config dictionary, current_run_dir, dan last_run_dir.
3. Jalankan uji kecil, kemudian periksa risiko berikut: widget baru tidak dimasukkan ke snapshot sehingga hasil tidak mencerminkan tampilan.
4. Terapkan validasi: bandingkan semua kunci UI dengan config.json dan uji perubahan saat job berjalan.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: anggap config.json sebagai kontrak job yang tidak boleh berubah selama eksekusi.

Worker Thread dan Stop Aman

Worker Thread dan Stop Aman merupakan bagian penting dari alur komputasi karena komputasi dipindahkan ke thread daemon agar UI tetap responsif dan penghentian dilakukan melalui checkpoint. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan worker thread dan stop aman sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada start_background_job, threading.Event, job_worker, callback SCF, dan request_stop. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah jangan mematikan proses secara paksa sebelum hasil parsial dan file ditulis dengan aman. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah backend eksternal tidak merespons event sampai subprocess mencapai titik polling. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah mengirim Stop pada fase berbeda dan memeriksa konsistensi file hasil parsial. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Job multi-molekul dihentikan setelah molekul ketiga; tiga hasil tetap tersimpan dan status akhir jelas. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk worker thread dan stop aman dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa start_background_job, threading.Event, job_worker, callback SCF, dan request_stop.
3. Jalankan uji kecil, kemudian periksa risiko berikut: backend eksternal tidak merespons event sampai subprocess mencapai titik polling.
4. Terapkan validasi: mengirim Stop pada fase berbeda dan memeriksa konsistensi file hasil parsial.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** jangan mematikan proses secara paksa sebelum hasil parsial dan file ditulis dengan aman.

Struktur Folder dan Penamaan Proyek

Pembahasan struktur folder dan penamaan proyek dimulai dari tujuan ilmiahnya: nama proyek, timestamp, dan UUID pendek membentuk folder unik untuk menghindari tabrakan hasil. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan safe_slug, run_id, config.json, input_manifest.json, dan direktori per molekul. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan nama proyek yang merepresentasikan eksperimen dan hindari mengganti nama folder setelah analisis. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena karakter ilegal, path terlalu panjang, atau dua proses menulis ke folder sama. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan uji nama dengan spasi dan simbol, serta pastikan semua path berada di bawah root yang disetujui. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Proyek 'Tamoxifen_ERa_B3LYP' menghasilkan folder unik yang mudah dicari. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk struktur folder dan penamaan proyek dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa safe_slug, run_id, config.json, input_manifest.json, dan direktori per molekul.
3. Jalankan uji kecil, kemudian periksa risiko berikut: karakter ilegal, path terlalu panjang, atau dua proses menulis ke folder sama.
4. Terapkan validasi: uji nama dengan spasi dan simbol, serta pastikan semua path berada di bawah root yang disetujui.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: gunakan nama proyek yang merepresentasikan eksperimen dan hindari mengganti nama folder setelah analisis.

Serialisasi Atomik dan Ketahanan Data

Serialisasi Atomik dan Ketahanan Data merupakan bagian penting dari alur komputasi karena penulisan JSON atomik mengurangi risiko file setengah tertulis ketika proses terhenti. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan serialisasi atomik dan ketahanan data sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada write_json_atomic, read_json_safe, _as_json_value, dan penyimpanan hasil. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah normalisasi tipe NumPy sebelum JSON dan jangan mengandalkan objek Python yang tidak serializable. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah NaN, array besar, atau exception saat rename meninggalkan data tidak konsisten. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah membuka semua JSON dengan parser standar dan menguji interupsi saat penulisan. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Listrik padam ketika katalog diperbarui; file lama tetap terbaca karena penggantian dilakukan atomik. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk serialisasi atomik dan ketahanan data dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa write_json_atomic, read_json_safe, _as_json_value, dan penyimpanan hasil.
3. Jalankan uji kecil, kemudian periksa risiko berikut: NaN, array besar, atau exception saat rename meninggalkan data tidak konsisten.
4. Terapkan validasi: membuka semua JSON dengan parser standar dan menguji interupsi saat penulisan.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: normalisasi tipe NumPy sebelum JSON dan jangan mengandalkan objek Python yang tidak serializable.

Gambar 4. Siklus job latar dan checkpoint penghentian aman pada aplikasi.

Gambar 5. Berkas utama yang harus dipertahankan sebagai paket reproduksibilitas.

BAB 5 — METODE ELEKTRONIK: HF, MP2, DAN DFT MODERN

Bab ini menjelaskan keluarga metode yang tersedia atau menjadi konteks aplikasi. Penekanan diberikan pada keadaan referensi, korelasi elektron, pilihan fungsional, dan hubungan antara metode dengan sifat yang dihitung.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Hartree–Fock dan Medan Rata-rata

Hartree–Fock dan Medan Rata-rata merupakan bagian penting dari alur komputasi karena HF mendekati fungsi gelombang dengan satu determinan dan memasukkan pertukaran eksak tanpa korelasi elektron dinamis. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan hartree–fock dan medan rata-rata sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada pilihan HF, RHF, ROHF, UHF dan logika Auto berdasarkan keadaan elektron. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah gunakan RHF untuk closed-shell, sedangkan open-shell memerlukan ROHF atau UHF dengan pemeriksaan kontaminasi spin. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah UHF dapat mematahkan simetri spin dan menghasilkan ⟨S²⟩ yang menyimpang. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah memeriksa konvergensi, stabilitas orbital, ⟨S²⟩, dan sensitivitas tebakan awal. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Radikal metil dihitung dengan UHF dan nilai ⟨S²⟩ dibandingkan dengan nilai ideal doublet. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk hartree–fock dan medan rata-rata dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa pilihan HF, RHF, ROHF, UHF dan logika Auto berdasarkan keadaan elektron.
3. Jalankan uji kecil, kemudian periksa risiko berikut: UHF dapat mematahkan simetri spin dan menghasilkan ⟨S²⟩ yang menyimpang.
4. Terapkan validasi: memeriksa konvergensi, stabilitas orbital, ⟨S²⟩, dan sensitivitas tebakan awal.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: gunakan RHF untuk closed-shell, sedangkan open-shell memerlukan ROHF atau UHF dengan pemeriksaan kontaminasi spin.

MP2 dan Korelasi Pasca-HF

Pembahasan mp2 dan korelasi pasca-hf dimulai dari tujuan ilmiahnya: MP2 menambahkan koreksi orde kedua terhadap referensi HF dan peka terhadap basis serta gap orbital. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan keluarga MP2 dalam make_pyscf_method dan pembatasan job frekuensi pada antarmuka. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan MP2 untuk sistem yang sesuai dan cek basis-set convergence serta kemungkinan near-degeneracy. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena MP2 dapat berperilaku buruk pada ikatan putus, logam transisi, atau sistem dengan korelasi statik. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan membandingkan energi korelasi, natural occupation bila tersedia, dan hasil DFT atau metode lebih tinggi. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Energi interaksi dimer kecil dihitung pada MP2 dengan basis beraugmen dan koreksi BSSE dipertimbangkan. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk mp2 dan korelasi pasca-hf dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa keluarga MP2 dalam make_pyscf_method dan pembatasan job frekuensi pada antarmuka.
3. Jalankan uji kecil, kemudian periksa risiko berikut: MP2 dapat berperilaku buruk pada ikatan putus, logam transisi, atau sistem dengan korelasi statik.
4. Terapkan validasi: membandingkan energi korelasi, natural occupation bila tersedia, dan hasil DFT atau metode lebih tinggi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan MP2 untuk sistem yang sesuai dan cek basis-set convergence serta kemungkinan near-degeneracy.

Kohn–Sham DFT dan Fungsional

Kohn–Sham DFT dan Fungsional merupakan bagian penting dari alur komputasi karena DFT memetakan masalah banyak elektron ke kerapatan elektron dengan energi pertukaran–korelasi hampiran. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan kohn–sham dft dan fungsional sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada dft.RKS/UKS, XC string, grid level, density fitting, dan daftar fungsional PySCF. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah pilih fungsional berdasarkan jenis sifat dan benchmark relevan, bukan popularitas semata. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah kesalahan delokalisasi, self-interaction, korelasi statik, dan ketergantungan grid. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah membandingkan beberapa kelas fungsional dan menguji grid serta basis. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Distribusi muatan donor–akseptor dibandingkan antara B3LYP dan fungsional range-separated. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk kohn–sham dft dan fungsional dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa dft.RKS/UKS, XC string, grid level, density fitting, dan daftar fungsional PySCF.
3. Jalankan uji kecil, kemudian periksa risiko berikut: kesalahan delokalisasi, self-interaction, korelasi statik, dan ketergantungan grid.
4. Terapkan validasi: membandingkan beberapa kelas fungsional dan menguji grid serta basis.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: pilih fungsional berdasarkan jenis sifat dan benchmark relevan, bukan popularitas semata.

B3LYP, PBE, dan PBE0

Pembahasan b3lyp, pbe, dan pbe0 dimulai dari tujuan ilmiahnya: fungsional GGA dan hybrid memberi keseimbangan berbeda antara exchange DFT dan exact exchange. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan default B3LYP serta pilihan PBE dan PBE0 pada backend yang didukung. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan default untuk pembelajaran tetapi lakukan benchmark untuk energi reaksi, logam, dan interaksi nonkovalen. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena B3LYP tanpa dispersi dapat melemahkan interaksi yang didominasi dispersi. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan menambahkan koreksi dispersi bila tersedia dan membandingkan geometri serta energi relatif. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Kompleks π-stacking diuji dengan protokol yang memasukkan dispersi, bukan B3LYP polos saja. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk b3lyp, pbe, dan pbe0 dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa default B3LYP serta pilihan PBE dan PBE0 pada backend yang didukung.
3. Jalankan uji kecil, kemudian periksa risiko berikut: B3LYP tanpa dispersi dapat melemahkan interaksi yang didominasi dispersi.
4. Terapkan validasi: menambahkan koreksi dispersi bila tersedia dan membandingkan geometri serta energi relatif.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan default untuk pembelajaran tetapi lakukan benchmark untuk energi reaksi, logam, dan interaksi nonkovalen.

M06-2X dan Fungsional Meta-hybrid

M06-2X dan Fungsional Meta-hybrid merupakan bagian penting dari alur komputasi karena meta-hybrid memakai informasi tambahan seperti rapat energi kinetik dan fraksi exact exchange yang lebih besar. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan m06-2x dan fungsional meta-hybrid sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada M06-2X tersedia pada daftar metode PySCF/ORCA sesuai dukungan backend. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah pertimbangkan M06-2X untuk termokimia dan nonkovalen organik, dengan grid cukup rapat. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah meta-GGA dapat lebih sensitif terhadap integrasi numerik dan geometri awal. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menaikkan grid level dan memeriksa kestabilan energi serta gradien. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Barrier reaksi organik dihitung ulang pada grid lebih rapat untuk memastikan perubahan kurang dari ambang penelitian. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk m06-2x dan fungsional meta-hybrid dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa M06-2X tersedia pada daftar metode PySCF/ORCA sesuai dukungan backend.
3. Jalankan uji kecil, kemudian periksa risiko berikut: meta-GGA dapat lebih sensitif terhadap integrasi numerik dan geometri awal.
4. Terapkan validasi: menaikkan grid level dan memeriksa kestabilan energi serta gradien.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: pertimbangkan M06-2X untuk termokimia dan nonkovalen organik, dengan grid cukup rapat.

CAM-B3LYP dan Range Separation

Pembahasan cam-b3lyp dan range separation dimulai dari tujuan ilmiahnya: range-separated hybrid meningkatkan exact exchange pada jarak jauh dan sering relevan untuk eksitasi transfer muatan. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan CAM-B3LYP tersedia pada daftar metode dan dapat digunakan bersama UV–Vis bila backend mendukung. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan ketika keadaan eksitasi memiliki karakter charge-transfer atau orbital terpisah ruang. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena spektrum dapat bergeser dan tidak ada satu fungsional yang universal untuk semua kromofor. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan membandingkan karakter orbital transisi, oscillator strength, dan data eksperimen. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Dye donor–π–akseptor dianalisis dengan B3LYP dan CAM-B3LYP untuk menilai sensitivitas eksitasi CT. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk cam-b3lyp dan range separation dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa CAM-B3LYP tersedia pada daftar metode dan dapat digunakan bersama UV–Vis bila backend mendukung.
3. Jalankan uji kecil, kemudian periksa risiko berikut: spektrum dapat bergeser dan tidak ada satu fungsional yang universal untuk semua kromofor.
4. Terapkan validasi: membandingkan karakter orbital transisi, oscillator strength, dan data eksperimen.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan ketika keadaan eksitasi memiliki karakter charge-transfer atau orbital terpisah ruang.

Tabel 4. Perbandingan ringkas keluarga metode

Metode	Korelasi	Kekuatan	Keterbatasan
HF	Pertukaran eksak; tanpa korelasi dinamis	Referensi jelas	Energi korelasi hilang
MP2	Orde kedua pasca-HF	Berguna pada banyak sistem weak-correlation	Peka basis dan near-degeneracy
GGA DFT	Fungsional kerapatan	Efisien	Self-interaction/delokalisasi
Hybrid DFT	DFT + exact exchange	Serbaguna	Biaya dan ketergantungan sistem
Range-separated	Exact exchange meningkat jarak jauh	Eksitasi CT	Perlu benchmark

BAB 6 — BASIS SET, KONVERGENSI SCF, GRID, MEMORI, DAN GPU

Bab ini membahas keputusan numerik yang menentukan biaya dan mutu hasil. Daftar aplikasi mencakup basis minimal, Pople, def2, correlation-consistent, augmented, dan polarization-consistent, serta kontrol toleransi SCF, grid DFT, thread, memori, density fitting, dan GPU4PySCF.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Makna Fungsi Basis

Pembahasan makna fungsi basis dimulai dari tujuan ilmiahnya: basis set membatasi ruang satu-elektron yang digunakan untuk merepresentasikan orbital molekul. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan BASIS_GROUPS dan BASIS_SETS yang dikelompokkan small sampai very large. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, mulai dengan double-zeta berpolarisasi untuk eksplorasi dan naikkan ke triple-zeta untuk hasil produksi yang sensitif. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena basis terlalu kecil memberi error basis-set yang tersamar sebagai efek metode. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan menguji perubahan energi relatif dan properti saat ukuran basis dinaikkan. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Geometri dioptimasi def2-SVP lalu energi tunggal dihitung def2-TZVP pada struktur yang sama. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk makna fungsi basis dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa BASIS_GROUPS dan BASIS_SETS yang dikelompokkan small sampai very large.
3. Jalankan uji kecil, kemudian periksa risiko berikut: basis terlalu kecil memberi error basis-set yang tersamar sebagai efek metode.
4. Terapkan validasi: menguji perubahan energi relatif dan properti saat ukuran basis dinaikkan.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: mulai dengan double-zeta berpolarisasi untuk eksplorasi dan naikkan ke triple-zeta untuk hasil produksi yang sensitif.

Polarisasi dan Fungsi Difus

Polarisasi dan Fungsi Difus merupakan bagian penting dari alur komputasi karena fungsi polarisasi memberi fleksibilitas sudut, sedangkan fungsi difus memperluas ekor kerapatan elektron. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan polarisasi dan fungsi difus sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada tanda bintang Pople, def2-SVPD/TZVPD, dan aug-cc-pVXZ. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah gunakan fungsi difus untuk anion, Rydberg, ikatan lemah, polarizabilitas, dan keadaan eksitasi tertentu. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah fungsi difus dapat menimbulkan ketergantungan linear dan biaya tinggi. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah memeriksa kondisi overlap, konvergensi SCF, dan sensitivitas properti. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Afinitas elektron dihitung dengan dan tanpa fungsi difus untuk menunjukkan dampaknya. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk polarisasi dan fungsi difus dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa tanda bintang Pople, def2-SVPD/TZVPD, dan aug-cc-pVXZ.
3. Jalankan uji kecil, kemudian periksa risiko berikut: fungsi difus dapat menimbulkan ketergantungan linear dan biaya tinggi.
4. Terapkan validasi: memeriksa kondisi overlap, konvergensi SCF, dan sensitivitas properti.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan fungsi difus untuk anion, Rydberg, ikatan lemah, polarizabilitas, dan keadaan eksitasi tertentu.

Keluarga def2 dan Correlation-consistent

Pembahasan keluarga def2 dan correlation-consistent dimulai dari tujuan ilmiahnya: def2 dirancang seimbang lintas tabel periodik, sedangkan cc-pVXZ memberi urutan sistematik untuk korelasi. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan def2-SVP sampai def2-QZVPP serta cc-pVDZ sampai cc-pVQZ. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, pilih keluarga yang konsisten dengan metode, unsur, dan tujuan extrapolation. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena mencampur kualitas basis antar atom tanpa alasan atau lupa ECP pada unsur berat. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan mendokumentasikan basis per unsur dan memeriksa dukungan backend. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Seri organik memakai def2-TZVP konsisten, sedangkan studi MP2 melakukan uji cc-pVDZ/cc-pVTZ. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk keluarga def2 dan correlation-consistent dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa def2-SVP sampai def2-QZVPP serta cc-pVDZ sampai cc-pVQZ.
3. Jalankan uji kecil, kemudian periksa risiko berikut: mencampur kualitas basis antar atom tanpa alasan atau lupa ECP pada unsur berat.
4. Terapkan validasi: mendokumentasikan basis per unsur dan memeriksa dukungan backend.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: pilih keluarga yang konsisten dengan metode, unsur, dan tujuan extrapolation.

Toleransi dan Siklus SCF

Toleransi dan Siklus SCF merupakan bagian penting dari alur komputasi karena SCF adalah iterasi nonlinear yang berhenti ketika perubahan energi atau residual memenuhi kriteria. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan toleransi dan siklus scf sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada toleransi 10^{-3} sampai 10^{-10}, max_scf_cycle, callback, dan fallback. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah gunakan toleransi ketat untuk gradien, frekuensi, dan perbedaan energi kecil. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah SCF konvergen ke solusi salah atau osilasi tanpa mencapai keadaan stabil. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menguji tebakan, damping/DIIS, level shift, dan stabilitas solusi. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Perhitungan frekuensi diulang dengan toleransi lebih ketat karena noise Hessian menghasilkan mode kecil negatif. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk toleransi dan siklus scf dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa toleransi 10^{-3} sampai 10^{-10}, max_scf_cycle, callback, dan fallback.
3. Jalankan uji kecil, kemudian periksa risiko berikut: SCF konvergen ke solusi salah atau osilasi tanpa mencapai keadaan stabil.
4. Terapkan validasi: menguji tebakan, damping/DIIS, level shift, dan stabilitas solusi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan toleransi ketat untuk gradien, frekuensi, dan perbedaan energi kecil.

Grid DFT dan Density Fitting

Pembahasan grid dft dan density fitting dimulai dari tujuan ilmiahnya: DFT memerlukan integrasi numerik exchange–correlation, sedangkan density fitting mengaproksimasi integral untuk efisiensi. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan grid_level 1–9 dan opsi density_fitting pada PySCF. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, naikkan grid untuk meta-GGA atau energi yang sensitif dan validasi efek RI pada target. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena grid kasar menghasilkan noise gradien atau perubahan energi yang tidak mulus. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan mengulang titik representatif pada grid lebih tinggi dan membandingkan angka signifikan. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Optimasi M06-2X yang berosilasi diuji ulang dengan grid lebih rapat. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk grid dft dan density fitting dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa grid_level 1–9 dan opsi density_fitting pada PySCF.
3. Jalankan uji kecil, kemudian periksa risiko berikut: grid kasar menghasilkan noise gradien atau perubahan energi yang tidak mulus.
4. Terapkan validasi: mengulang titik representatif pada grid lebih tinggi dan membandingkan angka signifikan.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: naikkan grid untuk meta-GGA atau energi yang sensitif dan validasi efek RI pada target.

CPU, Memori, dan GPU4PySCF

CPU, Memori, dan GPU4PySCF merupakan bagian penting dari alur komputasi karena kinerja ditentukan oleh algoritma, ukuran basis, memori, thread, I/O, dan dukungan akselerator. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan cpu, memori, dan gpu4pyscf sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada lib.num_threads, max_memory, deteksi CUDA, gpu_threads, dan konversi GPU4PySCF opsional. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah ukur waktu dan memori pada molekul representatif sebelum produksi skala besar. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah oversubscription thread, swapping, atau menganggap GPU selalu mempercepat semua modul. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah benchmark CPU vs GPU dengan hasil numerik yang setara dalam toleransi. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Job DFT besar diuji pada 2, 4, dan 8 thread untuk menemukan titik efisiensi terbaik. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk cpu, memori, dan gpu4pyscf dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa lib.num_threads, max_memory, deteksi CUDA, gpu_threads, dan konversi GPU4PySCF opsional.
3. Jalankan uji kecil, kemudian periksa risiko berikut: oversubscription thread, swapping, atau menganggap GPU selalu mempercepat semua modul.
4. Terapkan validasi: benchmark CPU vs GPU dengan hasil numerik yang setara dalam toleransi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** ukur waktu dan memori pada molekul representatif sebelum produksi skala besar.

Tabel 5. Kelompok basis set pada aplikasi

Kelompok	Contoh	Penggunaan praktis
Small	STO-3G, 3-21G, def2-SVP, cc-pVDZ	Eksplorasi dan geometri awal
Medium	def2-TZVP, cc-pVTZ, aug-cc-pVDZ	Produksi umum dan uji konvergensi
Large	def2-QZVP, cc-pVQZ, aug-cc-pVTZ	Properti sensitif dan benchmark
Very large	def2-QZVPP/D	Sistem kecil dengan kebutuhan tinggi

BAB 7 — OPTIMASI GEOMETRI, FREKUENSI, IR, DAN TERMOKIMIA

Bab ini memisahkan dengan tegas optimasi, analisis frekuensi, spektrum IR, dan termokimia. Pemisahan ini sesuai perubahan v2.3.4: IR dan Thermodynamics merupakan add-on yang berbeda, walaupun keduanya dapat memanfaatkan Hessian dan mode vibrasi.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Energi Potensial dan Gradien

Energi Potensial dan Gradien merupakan bagian penting dari alur komputasi karena optimasi mencari titik stasioner pada permukaan energi dengan menggunakan energi dan turunan terhadap koordinat. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan energi potensial dan gradien sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada job Geometry Optimization, gradient, callback, dan tabel riwayat energi-geometri. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah gunakan geometri awal masuk akal dan kriteria konvergensi sesuai ketelitian properti lanjutan. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah optimizer berhenti pada titik saddle, struktur terdisosiasi, atau minimum lokal yang tidak relevan. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah memeriksa gradien akhir, geometri, dan analisis frekuensi. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Konformer dengan energi force field rendah dioptimasi dan dibandingkan setelah semua gradien kecil. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk energi potensial dan gradien dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa job Geometry Optimization, gradient, callback, dan tabel riwayat energi-geometri.
3. Jalankan uji kecil, kemudian periksa risiko berikut: optimizer berhenti pada titik saddle, struktur terdisosiasi, atau minimum lokal yang tidak relevan.
4. Terapkan validasi: memeriksa gradien akhir, geometri, dan analisis frekuensi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: gunakan geometri awal masuk akal dan kriteria konvergensi sesuai ketelitian properti lanjutan.

geomeTRIC, PyBerny, dan Optimizer Lain

Pembahasan geometric, pyberny, dan optimizer lain dimulai dari tujuan ilmiahnya: optimizer mengubah koordinat berdasarkan gradien, sistem koordinat internal, dan aturan trust radius. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan geomeTRIC melalui PySCF, PyBerny, serta opsi ASE-BFGS. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, mulai dengan geomeTRIC dan gunakan alternatif bila topologi atau koordinat menyebabkan kegagalan. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena perbedaan optimizer disalahartikan sebagai perbedaan metode elektronik. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan membandingkan energi dan struktur akhir dari dua optimizer pada kasus sulit. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Molekul cincin fleksibel gagal di satu optimizer lalu berhasil dengan koordinat internal alternatif. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk geometric, pyberny, dan optimizer lain dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa geomeTRIC melalui PySCF, PyBerny, serta opsi ASE-BFGS.
3. Jalankan uji kecil, kemudian periksa risiko berikut: perbedaan optimizer disalahartikan sebagai perbedaan metode elektronik.
4. Terapkan validasi: membandingkan energi dan struktur akhir dari dua optimizer pada kasus sulit.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** mulai dengan geomeTRIC dan gunakan alternatif bila topologi atau koordinat menyebabkan kegagalan.

Hessian dan Mode Normal

Hessian dan Mode Normal merupakan bagian penting dari alur komputasi karena Hessian adalah matriks turunan kedua energi yang setelah pembobotan massa menghasilkan frekuensi dan vektor mode normal. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan hessian dan mode normal sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada calculate_pyscf_frequencies dan konversi Hessian ke cm^{-1}. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah hitung frekuensi pada tingkat teori yang konsisten dengan geometri bila biaya memungkinkan. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah translasi/rotasi tidak terproyeksi sempurna atau noise numerik menghasilkan mode kecil negatif. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menghitung 3N−6 atau 3N−5 mode, memvisualisasi vektor, dan menilai besar frekuensi imajiner. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Molekul nonlinier dengan N atom diharapkan memiliki 3N−6 mode vibrasi. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk hessian dan mode normal dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa calculate_pyscf_frequencies dan konversi Hessian ke cm^{-1}.
3. Jalankan uji kecil, kemudian periksa risiko berikut: translasi/rotasi tidak terproyeksi sempurna atau noise numerik menghasilkan mode kecil negatif.
4. Terapkan validasi: menghitung 3N−6 atau 3N−5 mode, memvisualisasi vektor, dan menilai besar frekuensi imajiner.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: hitung frekuensi pada tingkat teori yang konsisten dengan geometri bila biaya memungkinkan.

Spektrum IR sebagai Add-on Terpisah

Pembahasan spektrum ir sebagai add-on terpisah dimulai dari tujuan ilmiahnya: frekuensi harmonik dan intensitas IR menggambarkan perubahan momen dipol sepanjang koordinat normal. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan ADDON_OPTIONS memisahkan IR dari Thermodynamics dan panel hasil melaporkan frekuensi vibrasi. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan faktor skala yang sesuai level teori bila membandingkan dengan frekuensi fundamental eksperimen. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena menganggap frekuensi harmonik mentah identik dengan puncak eksperimen. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan membandingkan pola pita, intensitas relatif, faktor skala terdokumentasi, dan fase sampel. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Pita C=O dihitung, diskalakan berdasarkan sumber yang sesuai, lalu dibandingkan dengan FTIR. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk spektrum ir sebagai add-on terpisah dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa ADDON_OPTIONS memisahkan IR dari Thermodynamics dan panel hasil melaporkan frekuensi vibrasi.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menganggap frekuensi harmonik mentah identik dengan puncak eksperimen.
4. Terapkan validasi: membandingkan pola pita, intensitas relatif, faktor skala terdokumentasi, dan fase sampel.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan faktor skala yang sesuai level teori bila membandingkan dengan frekuensi fundamental eksperimen.

Termokimia pada T dan P

Termokimia pada T dan P merupakan bagian penting dari alur komputasi karena ZPE, entalpi, entropi, dan energi bebas diperoleh dari energi elektronik dan fungsi partisi dengan asumsi model tertentu. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan termokimia pada t dan p sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada run_pyscf_thermochemistry, temperatur default 298,15 K dan tekanan 101325 Pa. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah laporkan asumsi gas ideal, rigid rotor, harmonic oscillator, serta perlakuan mode frekuensi rendah. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah entropi mode lunak berlebihan atau membandingkan fase gas dengan eksperimen larutan. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah memeriksa mode rendah, satuan, temperatur, tekanan, dan koreksi standar-state. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: ΔG reaksi larutan tidak diambil langsung dari termokimia gas tanpa koreksi dan model solvasi. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk termokimia pada t dan p dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa run_pyscf_thermochemistry, temperatur default 298,15 K dan tekanan 101325 Pa.
3. Jalankan uji kecil, kemudian periksa risiko berikut: entropi mode lunak berlebihan atau membandingkan fase gas dengan eksperimen larutan.
4. Terapkan validasi: memeriksa mode rendah, satuan, temperatur, tekanan, dan koreksi standar-state.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: laporkan asumsi gas ideal, rigid rotor, harmonic oscillator, serta perlakuan mode frekuensi rendah.

Frekuensi Imajiner dan Karakter Titik Stasioner

Pembahasan frekuensi imajiner dan karakter titik stasioner dimulai dari tujuan ilmiahnya: frekuensi negatif menandakan kelengkungan negatif sepanjang suatu mode pada pendekatan harmonik. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan imaginary_frequency_count dan peringatan hasil. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, minimum harus tidak memiliki mode imajiner signifikan; keadaan transisi idealnya memiliki satu mode yang relevan. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena mengabaikan mode imajiner atau menganggap semua mode kecil negatif sebagai keadaan transisi. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

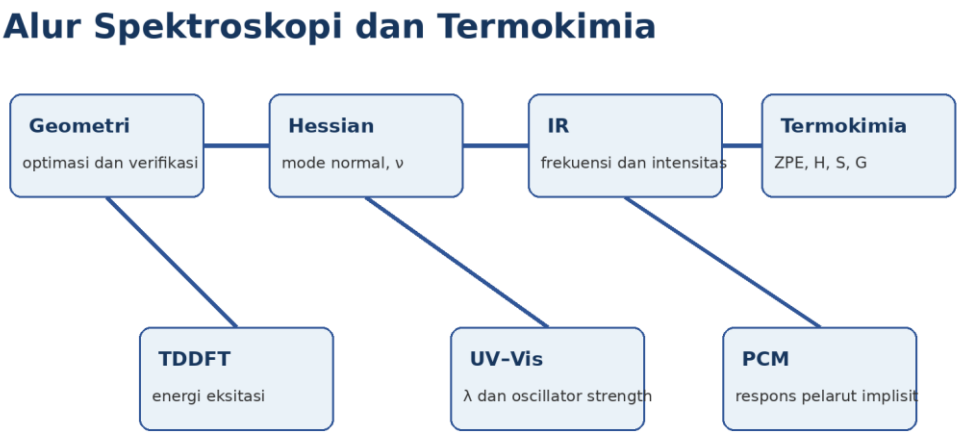
Bukti bahwa prosedur layak dipakai diperoleh dengan memvisualisasi mode, menggeser geometri sepanjang mode, dan mengoptimasi ulang. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Satu mode −450 cm^{-1} yang memutus/membentuk ikatan mendukung kandidat keadaan transisi. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk frekuensi imajiner dan karakter titik stasioner dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa imaginary_frequency_count dan peringatan hasil.
3. Jalankan uji kecil, kemudian periksa risiko berikut: mengabaikan mode imajiner atau menganggap semua mode kecil negatif sebagai keadaan transisi.
4. Terapkan validasi: memvisualisasi mode, menggeser geometri sepanjang mode, dan mengoptimasi ulang.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: minimum harus tidak memiliki mode imajiner signifikan; keadaan transisi idealnya memiliki satu mode yang relevan.



Gambar 6. Hubungan optimasi, Hessian, IR, termokimia, TDDFT, UV–Vis, dan PCM.

Tabel 6. Makna jumlah frekuensi imajiner

Temuan	Interpretasi awal	Tindakan
0	Kandidat minimum	Periksa mode sangat rendah dan geometri
1 signifikan	Kandidat keadaan transisi	Visualisasi mode dan IRC bila perlu
>1 signifikan	Saddle orde lebih tinggi	Geser geometri dan optimasi ulang
Kecil dekat nol	Noise/rotasi/konformer lunak	Ketatkan SCF/grid dan periksa mode

BAB 8 — UV–VIS, PCM, NMR, RAMAN, G3MP2, DAN NBO

Bab ini membahas add-on dan alur properti lanjutan. Tidak semua kombinasi tersedia pada setiap backend; aplikasi memakai adapter PySCF atau executable eksternal dan harus melaporkan keterbatasan tersebut secara eksplisit.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

TDDFT dan Eksitasi Elektronik

Pembahasan tddft dan eksitasi elektronik dimulai dari tujuan ilmiahnya: TDDFT linear-response menghitung energi eksitasi dan kekuatan osilator dari respons keadaan dasar. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan run_pyscf_uvvis dan parameter jumlah akar excited states. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, minta cukup banyak akar untuk mencakup rentang spektrum yang diminati. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena akar yang diinginkan berubah urutan atau memiliki karakter charge-transfer yang buruk pada fungsional tertentu. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan memeriksa komposisi transisi, oscillator strength, konvergensi akar, dan sensitivitas fungsional. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Dua puluh akar dihitung untuk memastikan pita UV dan awal daerah tampak tercakup. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk tddft dan eksitasi elektronik dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa run_pyscf_uvvis dan parameter jumlah akar excited states.
3. Jalankan uji kecil, kemudian periksa risiko berikut: akar yang diinginkan berubah urutan atau memiliki karakter charge-transfer yang buruk pada fungsional tertentu.
4. Terapkan validasi: memeriksa komposisi transisi, oscillator strength, konvergensi akar, dan sensitivitas fungsional.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** minta cukup banyak akar untuk mencakup rentang spektrum yang diminati.

Konversi Energi ke Panjang Gelombang

Konversi Energi ke Panjang Gelombang merupakan bagian penting dari alur komputasi karena hubungan λ(nm)=1239,841984/E(eV) menghubungkan energi foton dan panjang gelombang di vakum. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan konversi energi ke panjang gelombang sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada konstanta EV_NM dan tabel UV–Vis. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah jangan menafsirkan panjang gelombang tanpa oscillator strength dan karakter keadaan. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah membagi dengan energi sangat kecil atau mengurutkan pita hanya berdasarkan λ. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah memeriksa satuan, energi positif, dan konsistensi konversi. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Eksitasi 3,10 eV diperkirakan berada sekitar 400 nm sebelum memperhitungkan pelebaran spektrum. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk konversi energi ke panjang gelombang dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa konstanta EV_NM dan tabel UV–Vis.
3. Jalankan uji kecil, kemudian periksa risiko berikut: membagi dengan energi sangat kecil atau mengurutkan pita hanya berdasarkan λ.
4. Terapkan validasi: memeriksa satuan, energi positif, dan konsistensi konversi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** jangan menafsirkan panjang gelombang tanpa oscillator strength dan karakter keadaan.

PCM dan Efek Pelarut

Pembahasan pcm dan efek pelarut dimulai dari tujuan ilmiahnya: PCM merepresentasikan pelarut sebagai medium dielektrik kontinu yang merespons distribusi muatan solut. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan alur buku menempatkan PCM pada workflow UV–Vis/energi meskipun dukungan spesifik harus mengikuti backend dan kode yang digunakan. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, bedakan solvasi keadaan dasar, nonequilibrium excitation, dan kebutuhan molekul pelarut eksplisit. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena menganggap PCM menangkap ikatan hidrogen spesifik atau perubahan konformasi otomatis. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan membandingkan fase gas, PCM, dan cluster–continuum bila interaksi spesifik penting. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Kromofor berikatan hidrogen diuji dengan satu molekul pelarut eksplisit di dalam lingkungan kontinu. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk pcm dan efek pelarut dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa alur buku menempatkan PCM pada workflow UV–Vis/energi meskipun dukungan spesifik harus mengikuti backend dan kode yang digunakan.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menganggap PCM menangkap ikatan hidrogen spesifik atau perubahan konformasi otomatis.
4. Terapkan validasi: membandingkan fase gas, PCM, dan cluster–continuum bila interaksi spesifik penting.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: bedakan solvasi keadaan dasar, nonequilibrium excitation, dan kebutuhan molekul pelarut eksplisit.

NMR dan Shielding

NMR dan Shielding merupakan bagian penting dari alur komputasi karena pergeseran kimia NMR diperoleh dari shielding relatif terhadap standar dan sensitif terhadap geometri, konformer, serta pelarut. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan nmr dan shielding sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada run_pyscf_nmr melalui modul properti PySCF ketika tersedia. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah gunakan referensi konsisten dan lakukan averaging konformer bila perlu. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah membandingkan shielding absolut langsung dengan δ eksperimen. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menghitung standar pada tingkat teori sama dan mengevaluasi regresi terhadap data eksperimen. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Shielding ^{1}H senyawa dan TMS dihitung untuk memperoleh pergeseran relatif. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk nmr dan shielding dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa run_pyscf_nmr melalui modul properti PySCF ketika tersedia.
3. Jalankan uji kecil, kemudian periksa risiko berikut: membandingkan shielding absolut langsung dengan δ eksperimen.
4. Terapkan validasi: menghitung standar pada tingkat teori sama dan mengevaluasi regresi terhadap data eksperimen.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan referensi konsisten dan lakukan averaging konformer bila perlu.

Raman melalui ORCA

Pembahasan raman melalui orca dimulai dari tujuan ilmiahnya: aktivitas Raman berkaitan dengan perubahan polarizabilitas sepanjang mode normal. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan run_orca_raman_addon dan input temperatur/tekanan saat add-on Raman dipilih. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, pastikan ORCA tersedia dan metode/basis sesuai dengan target vibrasi. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena adapter eksternal gagal atau output versi baru tidak cocok dengan parser. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan menyimpan input/output ORCA dan membandingkan frekuensi dengan hasil utama. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Pita Raman simetris yang lemah di IR digunakan untuk melengkapi penugasan mode. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk raman melalui orca dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa run_orca_raman_addon dan input temperatur/tekanan saat add-on Raman dipilih.
3. Jalankan uji kecil, kemudian periksa risiko berikut: adapter eksternal gagal atau output versi baru tidak cocok dengan parser.
4. Terapkan validasi: menyimpan input/output ORCA dan membandingkan frekuensi dengan hasil utama.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: pastikan ORCA tersedia dan metode/basis sesuai dengan target vibrasi.

G3MP2 dan NBO sebagai Adapter Eksternal

G3MP2 dan NBO sebagai Adapter Eksternal merupakan bagian penting dari alur komputasi karena metode komposit dan analisis orbital alami memerlukan program serta lisensi yang berbeda dari jalur PySCF inti. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan g3mp2 dan nbo sebagai adapter eksternal sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada run_g3mp2_addon memakai Gaussian dan run_nbo_addon memakai ORCA + NBO6/7 dengan def2-TZPP. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah verifikasi lisensi, executable, versi, dan format output sebelum produksi. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah hasil add-on tidak tersedia tetapi dianggap nol atau sukses. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah periksa file keluaran asli, kode terminasi, dan nilai yang diparsing. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Analisis NBO hanya dilaporkan bila blok NBO benar-benar ada dan basis def2-TZPP digunakan sesuai adapter. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk g3mp2 dan nbo sebagai adapter eksternal dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa run_g3mp2_addon memakai Gaussian dan run_nbo_addon memakai ORCA + NBO6/7 dengan def2-TZPP.
3. Jalankan uji kecil, kemudian periksa risiko berikut: hasil add-on tidak tersedia tetapi dianggap nol atau sukses.
4. Terapkan validasi: periksa file keluaran asli, kode terminasi, dan nilai yang diparsing.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: verifikasi lisensi, executable, versi, dan format output sebelum produksi.

Tabel 7. Add-on, backend, dan prasyarat

Add-on	Jalur	Prasyarat/validasi
IR	PySCF/frekuensi	Hessian, minimum, faktor skala
Thermodynamics	PySCF thermo	T, P, mode rendah, standar-state
UV-Vis	PySCF TDDFT	Jumlah akar, f, karakter transisi
NMR	PySCF prop	Modul properti dan standar
Raman	Adapter ORCA	Executable dan parser
ΔHf G3MP2	Adapter Gaussian	Lisensi dan terminasi normal
NBO	ORCA + NBO6/7	Lisensi; basis def2-TZPP

BAB 9 — HASIL, ORBITAL, MEP, DESKRIPTOR, DAN EKSPOR

Bab ini mengajarkan pembacaan hasil secara berlapis: identitas dan struktur, energi, geometri, orbital, muatan, gradien, spektrum, deskriptor, hingga file volumetrik. Setiap tabel harus dikaitkan dengan kualitas job sumber.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Ringkasan Energi Multi-satuan

Ringkasan Energi Multi-satuan merupakan bagian penting dari alur komputasi karena konversi satuan memudahkan komunikasi tetapi tidak menambah informasi fisik baru. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan ringkasan energi multi-satuan sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada energy_units memakai konstanta Hartree ke eV, kJ mol^{-1}, kcal mol^{-1}, dan cm^{-1}. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah simpan Hartree sebagai nilai utama dan bulatkan konversi sesuai ketidakpastian model. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah terlalu banyak angka signifikan memberi kesan akurasi palsu. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah menghitung ulang satu konversi dan memeriksa tanda serta satuan. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Perbedaan energi 0,001 Hartree dilaporkan sekitar 2,6255 kJ mol^{-1}, bukan sebagai dua energi total yang sangat besar. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk ringkasan energi multi-satuan dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa energy_units memakai konstanta Hartree ke eV, kJ mol^{-1}, kcal mol^{-1}, dan cm^{-1}.
3. Jalankan uji kecil, kemudian periksa risiko berikut: terlalu banyak angka signifikan memberi kesan akurasi palsu.
4. Terapkan validasi: menghitung ulang satu konversi dan memeriksa tanda serta satuan.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: simpan Hartree sebagai nilai utama dan bulatkan konversi sesuai ketidakpastian model.

HOMO, LUMO, dan Gap

Pembahasan homo, lumo, dan gap dimulai dari tujuan ilmiahnya: energi orbital Kohn–Sham atau HF berguna untuk analisis kualitatif tetapi tidak selalu sama dengan potensial ionisasi atau afinitas elektron. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan calculate_frontier_orbitals, orbital_dataframe, dan cube HOMO/LUMO. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan gap untuk tren dalam protokol konsisten dan hindari klaim eksperimental langsung. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena menafsirkan gap orbital sebagai band gap optik tanpa TDDFT atau ΔSCF. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan membandingkan dengan energi eksitasi dan memvisualisasi karakter orbital. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Substituen donor menurunkan gap seri molekul, lalu tren diperiksa dengan TDDFT. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk homo, lumo, dan gap dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa calculate_frontier_orbitals, orbital_dataframe, dan cube HOMO/LUMO.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menafsirkan gap orbital sebagai band gap optik tanpa TDDFT atau ΔSCF.
4. Terapkan validasi: membandingkan dengan energi eksitasi dan memvisualisasi karakter orbital.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** gunakan gap untuk tren dalam protokol konsisten dan hindari klaim eksperimental langsung.

Momen Dipol dan Muatan Mulliken

Momen Dipol dan Muatan Mulliken merupakan bagian penting dari alur komputasi karena momen dipol adalah properti total, sedangkan muatan atom merupakan partisi model-dependent. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan momen dipol dan muatan mulliken sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada vektor/magnitudo dipol dan mulliken_dataframe. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah bandingkan muatan hanya dengan skema sama, basis sama, dan geometri sebanding. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah menganggap muatan Mulliken sebagai observabel unik dan stabil terhadap basis. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah membandingkan tren dengan skema lain atau MEP dan memeriksa jumlah muatan total. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Muatan atom karbonil digunakan secara relatif antar analog, bukan sebagai muatan absolut yang eksak. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk momen dipol dan muatan mulliken dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa vektor/magnitudo dipol dan mulliken_dataframe.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menganggap muatan Mulliken sebagai observabel unik dan stabil terhadap basis.
4. Terapkan validasi: membandingkan tren dengan skema lain atau MEP dan memeriksa jumlah muatan total.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: bandingkan muatan hanya dengan skema sama, basis sama, dan geometri sebanding.

MEP, Density, dan Gaussian Cube

Pembahasan mep, density, dan gaussian cube dimulai dari tujuan ilmiahnya: file cube menyimpan nilai volumetrik pada grid untuk visualisasi kerapatan, potensial elektrostatik, dan orbital. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan generate_pyscf_cube_files serta cube_grid default 60 atau 216.000 titik per cube. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, pilih grid dan bounding box yang memadai tanpa memboroskan memori. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena membandingkan warna MEP dari skala visual berbeda. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan gunakan isovalue dan skala warna konsisten serta periksa orientasi molekul. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Peta MEP tiga analog divisualkan dengan rentang warna sama untuk menilai daerah elektrofilik. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk mep, density, dan gaussian cube dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa generate_pyscf_cube_files serta cube_grid default 60 atau 216.000 titik per cube.
3. Jalankan uji kecil, kemudian periksa risiko berikut: membandingkan warna MEP dari skala visual berbeda.
4. Terapkan validasi: gunakan isovalue dan skala warna konsisten serta periksa orientasi molekul.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** pilih grid dan bounding box yang memadai tanpa memboroskan memori.

Geometri: Ikatan, Sudut, dan Dihedral

Geometri: Ikatan, Sudut, dan Dihedral merupakan bagian penting dari alur komputasi karena parameter internal merangkum perubahan struktur dan membantu mendeteksi hasil tidak wajar. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan geometri: ikatan, sudut, dan dihedral sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada calculate_geometry_tables menghasilkan semua panjang ikatan, sudut, dan dihedral. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah pilih parameter yang relevan dengan hipotesis dan laporkan satuan. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah membandingkan indeks atom yang berubah antar format atau molekul. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah gunakan label atom konsisten dan cek langsung pada struktur 3D. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Sudut dihedral penghubung dua cincin dilacak sepanjang seri substituen. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk geometri: ikatan, sudut, dan dihedral dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa calculate_geometry_tables menghasilkan semua panjang ikatan, sudut, dan dihedral.
3. Jalankan uji kecil, kemudian periksa risiko berikut: membandingkan indeks atom yang berubah antar format atau molekul.
4. Terapkan validasi: gunakan label atom konsisten dan cek langsung pada struktur 3D.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: pilih parameter yang relevan dengan hipotesis dan laporkan satuan.

QSAR/QSPR RDKit dan Deskriptor QM

Pembahasan qsar/qspr rdkit dan deskriptor qm dimulai dari tujuan ilmiahnya: deskriptor 2D/3D dan QM dapat digabungkan untuk membangun representasi molekul, tetapi memerlukan kurasi dan validasi statistik. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan calculate_rdkit_qsar_qspr_descriptors dan add_qm_qsar_descriptors. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, hapus deskriptor konstan, redundan, atau bocor target sebelum pemodelan. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena jumlah deskriptor jauh melebihi jumlah molekul dan menyebabkan overfitting. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan split data, cross-validation, domain applicability, dan pengujian eksternal. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Seri 50 molekul tidak boleh digunakan melatih model dengan ratusan fitur tanpa regularisasi dan validasi ketat. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk qsar/qspr rdkit dan deskriptor qm dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa calculate_rdkit_qsar_qspr_descriptors dan add_qm_qsar_descriptors.
3. Jalankan uji kecil, kemudian periksa risiko berikut: jumlah deskriptor jauh melebihi jumlah molekul dan menyebabkan overfitting.
4. Terapkan validasi: split data, cross-validation, domain applicability, dan pengujian eksternal.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** hapus deskriptor konstan, redundan, atau bocor target sebelum pemodelan.

Ekspor MOL, SDF, PDB, XYZ, dan Log

Ekspor MOL, SDF, PDB, XYZ, dan Log merupakan bagian penting dari alur komputasi karena setiap format ekspor melayani tujuan berbeda dan tidak semuanya menyimpan bond order atau metadata sama. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan ekspor mol, sdf, pdb, xyz, dan log sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada export_bonded_formats, save_result_files, write_summary_log, dan tombol salin. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah arsipkan SDF/MOL untuk konektivitas, XYZ untuk koordinat sederhana, PDB untuk kompatibilitas visual, dan JSON/CSV untuk data. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah menggunakan XYZ sebagai satu-satunya arsip sehingga konektivitas hilang. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah membuka file pada program kedua dan membandingkan jumlah atom, ikatan, serta koordinat. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Struktur akhir dikirim ke PyMOL dalam PDB dan ke workflow cheminformatics dalam SDF. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk ekspor mol, sdf, pdb, xyz, dan log dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa export_bonded_formats, save_result_files, write_summary_log, dan tombol salin.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menggunakan XYZ sebagai satu-satunya arsip sehingga konektivitas hilang.
4. Terapkan validasi: membuka file pada program kedua dan membandingkan jumlah atom, ikatan, serta koordinat.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: arsipkan SDF/MOL untuk konektivitas, XYZ untuk koordinat sederhana, PDB untuk kompatibilitas visual, dan JSON/CSV untuk data.

Tabel 8. Format keluaran dan informasi yang dipertahankan

Format	Koordinat	Konektivitas	Data tambahan
XYZ	Ya	Tidak eksplisit	Komentar terbatas
MOL	Ya	Ya	Satu molekul
SDF	Ya	Ya	Properti/tag
PDB	Ya	Terbatas	Residue/atom naming
CUBE	Grid volumetrik	Bukan tujuan utama	Density/MEP/orbital

BAB 10 — WORKFLOW RISET, TROUBLESHOOTING, LATIHAN, DAN PENGEMBANGAN

Bab terakhir menyatukan seluruh komponen menjadi workflow penelitian yang dapat digunakan dalam praktikum, tugas akhir, dan riset. Bagian akhir memuat dua puluh soal latihan dengan pembahasan serta soal pengembangan.

Capaian pembelajaran bab ini adalah kemampuan menjelaskan konsep, mengoperasikan fitur terkait, mengidentifikasi risiko, dan merancang validasi. Pembaca dianjurkan menjalankan contoh dengan molekul kecil terlebih dahulu, menyimpan seluruh folder job, lalu membandingkan hasil dengan tabel dan kriteria yang disediakan.

Workflow Optimasi dan Properti Dasar

Pembahasan workflow optimasi dan properti dasar dimulai dari tujuan ilmiahnya: alur yang baik memisahkan persiapan, optimasi, verifikasi minimum, dan perhitungan properti. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan job optimization + frequency, add-on, tabel geometri, orbital, dan file final. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, gunakan geometri terverifikasi sebagai sumber semua properti yang dibandingkan. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena menghitung spektrum pada struktur awal atau geometri yang belum minimum. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan memeriksa frekuensi imajiner, perubahan geometri, dan konsistensi metode. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Paracetamol dioptimasi B3LYP/def2-SVP, diverifikasi frekuensi, lalu dianalisis HOMO–LUMO dan MEP. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk workflow optimasi dan properti dasar dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa job optimization + frequency, add-on, tabel geometri, orbital, dan file final.
3. Jalankan uji kecil, kemudian periksa risiko berikut: menghitung spektrum pada struktur awal atau geometri yang belum minimum.
4. Terapkan validasi: memeriksa frekuensi imajiner, perubahan geometri, dan konsistensi metode.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: gunakan geometri terverifikasi sebagai sumber semua properti yang dibandingkan.

Workflow Seri Molekul dan Benchmark

Workflow Seri Molekul dan Benchmark merupakan bagian penting dari alur komputasi karena perbandingan seri memerlukan protokol identik, penamaan stabil, dan pengendalian kegagalan individual. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan workflow seri molekul dan benchmark sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada multi-molecule job, manifest, unique_names, progress, dan hasil parsial. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah tetapkan molekul representatif untuk benchmark sebelum menjalankan seluruh seri. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah satu molekul gagal tetapi barisnya diabaikan sehingga dataset bias. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah buat tabel kelulusan job dan alasan eksklusi sebelum analisis statistik. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Lima puluh preset dihitung pada metode murah; hanya hasil yang lulus kriteria masuk tahap DFT. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk workflow seri molekul dan benchmark dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa multi-molecule job, manifest, unique_names, progress, dan hasil parsial.
3. Jalankan uji kecil, kemudian periksa risiko berikut: satu molekul gagal tetapi barisnya diabaikan sehingga dataset bias.
4. Terapkan validasi: buat tabel kelulusan job dan alasan eksklusi sebelum analisis statistik.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** tetapkan molekul representatif untuk benchmark sebelum menjalankan seluruh seri.

Diagnosis SCF dan Optimasi

Pembahasan diagnosis scf dan optimasi dimulai dari tujuan ilmiahnya: error numerik harus dilokalisasi ke input, keadaan elektron, SCF, optimizer, atau parser backend. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan fallback SCF, log callback, max cycles, toleransi, dan exception traceback. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, ubah satu faktor pada satu waktu dan simpan setiap percobaan sebagai job terpisah. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena mengubah banyak parameter sekaligus sehingga penyebab perbaikan tidak diketahui. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan membandingkan log siklus, energi akhir, dan struktur setelah setiap intervensi. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: SCF open-shell gagal; multiplicity diperiksa sebelum mencoba damping atau basis lebih kecil. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk diagnosis scf dan optimasi dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa fallback SCF, log callback, max cycles, toleransi, dan exception traceback.
3. Jalankan uji kecil, kemudian periksa risiko berikut: mengubah banyak parameter sekaligus sehingga penyebab perbaikan tidak diketahui.
4. Terapkan validasi: membandingkan log siklus, energi akhir, dan struktur setelah setiap intervensi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: ubah satu faktor pada satu waktu dan simpan setiap percobaan sebagai job terpisah.

Diagnosis Backend Eksternal dan Parser

Diagnosis Backend Eksternal dan Parser merupakan bagian penting dari alur komputasi karena job CLI terdiri dari penulisan input, peluncuran subprocess, pemantauan, dan parsing output. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan diagnosis backend eksternal dan parser sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada run_process_interruptible, parser energi/frekuensi MOPAC, serta adapter xTB dan ORCA. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah simpan output mentah dan jangan hanya mengandalkan pesan ringkas antarmuka. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah format output berubah antar versi sehingga parser tidak menemukan nilai meskipun program selesai. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah cocokkan regex dengan output resmi dan tambahkan fixture uji dari beberapa versi. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: MOPAC selesai tetapi energi tidak terparse; diagnostic excerpt dipakai menemukan label output yang berubah. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk diagnosis backend eksternal dan parser dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa run_process_interruptible, parser energi/frekuensi MOPAC, serta adapter xTB dan ORCA.
3. Jalankan uji kecil, kemudian periksa risiko berikut: format output berubah antar versi sehingga parser tidak menemukan nilai meskipun program selesai.
4. Terapkan validasi: cocokkan regex dengan output resmi dan tambahkan fixture uji dari beberapa versi.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

> **Keputusan praktis:** simpan output mentah dan jangan hanya mengandalkan pesan ringkas antarmuka.

Pelaporan Ilmiah dan Reproduksibilitas

Pembahasan pelaporan ilmiah dan reproduksibilitas dimulai dari tujuan ilmiahnya: bagian metode harus memuat versi, struktur awal, charge, multiplicity, metode, basis, konvergensi, grid, optimizer, model pelarut, dan koreksi. Banyak kegagalan studi komputasi bukan disebabkan program yang tidak bekerja, melainkan karena pertanyaan penelitian tidak diterjemahkan menjadi model yang tepat. Sebelum menjalankan job, peneliti harus menyebutkan observabel yang dibutuhkan, rentang akurasi yang masuk akal, dan sumber ketidakpastian dominan. Dengan kerangka ini, ukuran basis, metode korelasi, model pelarut, dan kedalaman optimasi dapat dipilih secara rasional.

Implementasi pada aplikasi berkaitan langsung dengan config, manifest, log, dan folder hasil menyediakan data untuk menyusun pelaporan. Komponen antarmuka mengubah pilihan pengguna menjadi objek konfigurasi yang dibekukan ketika tombol Run ditekan. Pendekatan snapshot ini penting karena perubahan widget setelah job dimulai tidak mengubah job yang sedang berjalan. Setiap hasil seharusnya dibaca bersama konfigurasi tersebut; tabel energi tanpa metadata metode dan basis tidak cukup untuk reproduksibilitas.

Untuk penggunaan rutin, buat tabel metode komputasi sebelum menulis interpretasi hasil. Strategi tersebut menghindari dua ekstrem: protokol terlalu murah yang gagal menangkap fenomena penting, atau protokol terlalu mahal yang tidak memberi peningkatan bermakna. Uji konvergensi kecil pada molekul representatif sering lebih informatif daripada langsung menghitung seluruh seri dengan pengaturan maksimum. Hasil uji itu kemudian menjadi dasar protokol produksi.

Perhatian khusus diperlukan karena laporan hanya menyebut 'menggunakan DFT' tanpa detail yang cukup. Kesalahan semacam ini dapat merambat ke seluruh keluaran, misalnya orbital, muatan, frekuensi, atau deskriptor turunan. Bila sumber masalah adalah struktur awal atau keadaan spin, memperketat toleransi SCF saja tidak akan menyelesaikannya. Diagnosis harus mengikuti rantai sebab: input, model elektronik, prosedur numerik, lalu interpretasi.

Bukti bahwa prosedur layak dipakai diperoleh dengan minta rekan mereproduksi satu hasil dari metode tertulis tanpa bantuan lisan. Pemeriksaan silang terhadap metode kedua, basis yang lebih besar, atau data eksperimen dapat digunakan sesuai tujuan. Yang terpenting, kriteria penerimaan ditetapkan sebelum melihat hasil yang diinginkan. Praktik ini mengurangi bias seleksi dan memperkuat argumentasi ketika hasil dibahas dalam laporan, skripsi, tesis, atau artikel.

Studi kasus dan langkah kerja

Skenario: Artikel melaporkan B3LYP/def2-SVP, grid, toleransi, suhu, tekanan, versi PySCF, dan prosedur frekuensi. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk pelaporan ilmiah dan reproduksibilitas dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa config, manifest, log, dan folder hasil menyediakan data untuk menyusun pelaporan.
3. Jalankan uji kecil, kemudian periksa risiko berikut: laporan hanya menyebut 'menggunakan DFT' tanpa detail yang cukup.
4. Terapkan validasi: minta rekan mereproduksi satu hasil dari metode tertulis tanpa bantuan lisan.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: buat tabel metode komputasi sebelum menulis interpretasi hasil.

Roadmap Pengembangan Aplikasi

Roadmap Pengembangan Aplikasi merupakan bagian penting dari alur komputasi karena pengembangan harus menjaga kompatibilitas fitur sambil menambah pengujian, keamanan path, dan dukungan properti. Dalam praktik, istilah ini tidak cukup dipahami sebagai nama menu atau metode; pengguna perlu melihat hubungan antara model kimia, representasi struktur, parameter numerik, dan sifat yang hendak ditafsirkan. Keputusan yang dibuat pada tahap ini akan memengaruhi biaya komputasi, kestabilan konvergensi, serta tingkat keyakinan terhadap kesimpulan. Oleh sebab itu, buku ini menempatkan roadmap pengembangan aplikasi sebagai keputusan ilmiah yang harus dicatat, bukan sekadar pilihan antarmuka.

Pada QM Komputasi V2.3.4, penerapannya terlihat pada arsitektur fungsi modular, backend adapters, session state, dan katalog persisten. Aplikasi berusaha menjaga keterlacakan dengan menyimpan konfigurasi dan hasil di folder job. Meskipun antarmuka menyederhanakan eksekusi, tanggung jawab validasi tetap berada pada peneliti: struktur, muatan, multiplicity, metode, basis, dan jenis job harus konsisten. Default yang nyaman berguna untuk pembelajaran, tetapi bukan jaminan bahwa semua sistem molekul dapat diperlakukan dengan protokol yang sama.

Keputusan operasional yang disarankan adalah prioritaskan perbaikan direktori data pengguna, test suite parser, environment lock, dan provenance versi. Pilihan tersebut perlu ditulis di bagian metode penelitian bersama versi perangkat lunak dan kondisi komputasi. Bila dilakukan perbandingan antar molekul, semua anggota seri harus menggunakan definisi energi, tingkat teori, dan kriteria konvergensi yang sebanding. Perubahan satu parameter di tengah seri dapat menghasilkan kecenderungan semu yang berasal dari protokol, bukan dari kimia molekul.

Risiko utama pada topik ini adalah fitur baru mengubah hasil lama tanpa versioning atau regression test. Gejala risiko tidak selalu berupa pesan error; hasil dapat terlihat rapi tetapi secara kimia keliru. Karena itu pemeriksaan tidak boleh berhenti pada status 'selesai'. Pengguna perlu membaca log, memeriksa struktur akhir, mengevaluasi besaran yang tidak wajar, dan membandingkan dengan literatur atau eksperimen yang relevan. Untuk sistem terbuka, bermuatan, fleksibel, atau mengandung unsur berat, pemeriksaan ini menjadi semakin penting.

Validasi minimum yang dianjurkan ialah buat unit test, integration test molekul kecil, dan golden files keluaran. Validasi yang baik bersifat bertingkat: mulai dari pemeriksaan identitas kimia, kemudian stabilitas numerik, lalu kewajaran kimia dan sensitivitas terhadap model. Hasil yang lolos semua tingkat tersebut lebih layak digunakan untuk interpretasi spektrum, reaktivitas, atau perbandingan energi. Catatan validasi perlu disertakan dalam arsip proyek agar orang lain dapat menelusuri alasan di balik setiap keputusan.

Studi kasus dan langkah kerja

Skenario: Versi berikutnya mengubah lokasi qm_runs tetapi menyediakan migrasi katalog dan dokumentasi kompatibilitas. Skenario ini dipakai untuk menghubungkan konsep dengan tindakan yang dapat diverifikasi. Setiap langkah harus menghasilkan bukti berupa konfigurasi, log, struktur, atau tabel; kesimpulan tidak boleh hanya didasarkan pada tampilan antarmuka.

1. Tentukan tujuan untuk roadmap pengembangan aplikasi dan tulis besaran yang akan dibandingkan.
2. Siapkan input serta konfigurasi yang sesuai; khusus aplikasi, perhatikan bahwa arsitektur fungsi modular, backend adapters, session state, dan katalog persisten.
3. Jalankan uji kecil, kemudian periksa risiko berikut: fitur baru mengubah hasil lama tanpa versioning atau regression test.
4. Terapkan validasi: buat unit test, integration test molekul kecil, dan golden files keluaran.
5. Arsipkan config.json, manifest, log, struktur akhir, dan tabel hasil; catat setiap penyimpangan dari protokol.

Keputusan praktis: prioritaskan perbaikan direktori data pengguna, test suite parser, environment lock, dan provenance versi.

Tabel 9. Checklist kelulusan hasil

Lapisan	Pertanyaan kelulusan
Input	Identitas, charge, multiplicity, dan geometri benar?
Numerik	SCF/optimizer konvergen dan log bersih?
Struktur	Ikatan, sudut, dan konformer masuk akal?
Stasioner	Frekuensi mendukung minimum/TS?
Model	Metode, basis, grid, dan pelarut memadai?
Interpretasi	Kesimpulan didukung sensitivitas dan pembanding?
Reproduksi	Folder job dan versi lengkap tersedia?

LATIHAN TERSTRUKTUR DAN PEMBAHASAN

Dua puluh soal berikut mencakup operasi aplikasi, konsep metode, interpretasi, serta troubleshooting. Nomor soal menggunakan daftar bernomor Word dan tidak dimasukkan ke daftar isi. Setiap pembahasan menjelaskan alasan, bukan hanya jawaban akhir.

Pertanyaan 1. Jelaskan perbedaan peran RDKit dan backend QM dalam aplikasi.

Pembahasan. RDKit menangani identitas kimia, konektivitas, visualisasi, penambahan hidrogen, embedding 3D, serta pre-optimasi force field. Energi dan properti kuantum dihitung oleh backend terpilih seperti PySCF, MOPAC, xTB, atau ORCA. Karena itu geometri awal RDKit bukan hasil QM dan harus dibedakan dari geometri akhir backend.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 2. Mengapa charge dan multiplicity harus diperiksa bersama jumlah elektron?

Pembahasan. Charge menentukan jumlah elektron total, sedangkan multiplicity = 2S+1 menentukan keadaan spin. Kombinasi yang tidak kompatibel dapat menimbulkan error atau memaksa keadaan elektronik yang salah. Paritas elektron memberi pemeriksaan awal: jumlah elektron genap lazimnya cocok dengan multiplicity ganjil, sedangkan jumlah elektron ganjil dengan multiplicity genap, meskipun keadaan aktual tetap perlu pertimbangan kimia.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 3. Kapan RHF, ROHF, dan UHF dipilih?

Pembahasan. RHF dipakai untuk closed-shell dengan pasangan elektron lengkap. ROHF menjaga orbital spasial tertentu bersama sambil menangani open-shell. UHF memberi orbital berbeda untuk spin alfa dan beta sehingga fleksibel, tetapi dapat mengalami kontaminasi spin. Pemilihan harus disertai pemeriksaan konvergensi dan $\langle S^2 \rangle$.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 4. Mengapa energi total dua molekul dengan komposisi berbeda tidak boleh dibandingkan secara naif?

Pembahasan. Energi total mengandung kontribusi semua elektron dan inti; molekul dengan lebih banyak atom biasanya memiliki energi lebih negatif. Perbandingan bermakna menggunakan energi reaksi, energi relatif

konformer/isomer dengan komposisi sama, atau skema termokimia yang seimbang pada tingkat teori sama.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 5. Apa fungsi analisis frekuensi setelah optimasi?

Pembahasan. Frekuensi mengkarakterisasi titik stasioner. Minimum yang baik tidak memiliki frekuensi imajiner signifikan, sedangkan keadaan transisi idealnya mempunyai satu mode imajiner yang sesuai koordinat reaksi. Frekuensi juga menyediakan ZPE dan kontribusi termal, tetapi memerlukan asumsi harmonik dan pemeriksaan mode rendah.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 6. Bedakan Add-on IR dan Thermodynamics pada v2.3.4.

Pembahasan. IR berfokus pada mode vibrasi harmonik dan, bila tersedia, intensitas yang terkait perubahan momen dipol. Thermodynamics menggunakan energi dan frekuensi untuk menghitung ZPE, entalpi, entropi, serta energi bebas pada temperatur dan tekanan tertentu. Keduanya berkaitan melalui Hessian tetapi tujuan dan interpretasinya berbeda.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 7. Mengapa faktor skala frekuensi diperlukan?

Pembahasan. Frekuensi harmonik dihitung dengan model elektronik dan asumsi harmonik yang tidak sempurna. Faktor skala empiris yang spesifik terhadap kombinasi metode–basis dapat mendekatkan frekuensi fundamental atau ZPE. Faktor harus dipilih dari sumber yang sesuai dan tujuan scaling harus disebutkan.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 8. Apa dampak fungsi difus pada anion dan keadaan Rydberg?

Pembahasan. Fungsi difus memberi fleksibilitas pada daerah jauh dari inti sehingga kerapatan elektron yang menyebar dapat direpresentasikan. Tanpanya, anion, polarizabilitas, interaksi lemah, dan keadaan Rydberg dapat sangat keliru. Namun fungsi difus juga dapat meningkatkan biaya dan masalah ketergantungan linear.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 9. Mengapa default B3LYP/def2-SVP tidak otomatis tepat untuk semua riset?

Pembahasan. Default adalah titik awal yang praktis, bukan validasi universal. Kinerja B3LYP berbeda antar sifat dan sistem; def2-SVP masih berukuran double-zeta. Energi reaksi sensitif, interaksi dispersi, charge-transfer, anion, unsur berat, dan logam transisi mungkin memerlukan fungsional, koreksi, basis, relativitas, atau model pelarut lain.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 10. Bagaimana menguji konvergensi basis set secara sederhana?

Pembahasan. Pilih beberapa sistem representatif dan hitung besaran target dengan dua atau tiga kualitas basis dalam keluarga yang konsisten, misalnya def2-SVP → def2-TZVP → def2-QZVP. Evaluasi perubahan energi relatif atau properti, bukan energi total saja, lalu tetapkan ambang yang relevan dengan tujuan penelitian.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 11. Apa arti grid level pada DFT?

Pembahasan. Exchange–correlation DFT dievaluasi dengan integrasi numerik. Grid lebih rapat biasanya mengurangi error integrasi tetapi meningkatkan biaya. Meta-GGA dan gradien dapat lebih sensitif. Validasi dilakukan dengan mengulang titik representatif pada grid lebih tinggi dan memeriksa kestabilan energi, gradien, serta geometri.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 12. Mengapa tombol Stop menggunakan checkpoint aman?

Pembahasan. Menghentikan proses secara paksa dapat merusak file atau meninggalkan state tidak konsisten. Event penghentian diperiksa pada siklus SCF, langkah optimasi, atau proses eksternal sehingga aplikasi dapat menyimpan hasil parsial dan status dengan lebih aman. Respons tidak selalu seketika karena bergantung pada titik polling.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 13. Jelaskan risiko RUNS_DIR = APP_DIR / 'qm_runs' pada paket /opt.

Pembahasan. Direktori /opt biasanya hanya dapat ditulis administrator. Ketika aplikasi biasa mencoba membuat qm_runs atau upload_molekul.json di APP_DIR, PermissionError dapat terjadi. Data berubah seharusnya ditempatkan pada direktori pengguna, misalnya ~/.local/share/...; paket/launcher harus memastikan lokasi tersebut digunakan tanpa menjalankan seluruh aplikasi sebagai root.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 14. Apa yang harus diperiksa ketika MOPAC selesai tetapi energi gagal diparsing?

Pembahasan. Pertama simpan dan baca output mentah serta kode keluar. Bandingkan label energi dengan pola parser, perhatikan versi MOPAC dan notasi angka D/E. Gunakan diagnostic excerpt untuk melihat konteks. Jangan mengganti nilai dengan nol; parser perlu diperbaiki atau metode parsing cadangan digunakan dengan uji fixture.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 15. Mengapa HOMO–LUMO gap bukan otomatis energi serapan UV–Vis?

Pembahasan. Gap orbital adalah selisih eigenvalue satu-elektron pada keadaan dasar, sedangkan serapan melibatkan keadaan tereksitasi, interaksi elektron–hole, relaksasi, dan selection rules. TDDFT atau metode excited-state diperlukan untuk energi eksitasi dan oscillator strength. Gap tetap berguna sebagai deskriptor tren jika protokol konsisten.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 16. Bagaimana menafsirkan muatan Mulliken secara benar?

Pembahasan. Muatan Mulliken adalah partisi populasi basis yang sangat bergantung pada basis dan bukan observabel unik. Gunakan terutama untuk tren dalam seri yang dihitung identik, cek jumlah muatan total, dan bandingkan dengan skema atau MEP lain bila kesimpulan sensitif.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 17. Apa tujuan file Gaussian cube?

Pembahasan. Cube menyimpan nilai volumetrik pada grid, misalnya kerapatan elektron, potensial elektrostatik, HOMO, atau LUMO. File dipakai untuk visualisasi isosurface dan peta warna. Perbandingan antar molekul harus memakai grid, isovalue, orientasi, dan skala warna yang konsisten.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 18. Mengapa QSAR/QSPR memerlukan validasi di luar perhitungan deskriptor?

Pembahasan. Deskriptor hanyalah representasi. Dataset kecil dengan fitur banyak mudah overfit; korelasi antar fitur, kebocoran target, pembagian data, domain applicability, dan validasi eksternal harus ditangani. Hasil QM yang gagal juga tidak boleh masuk dataset tanpa penandaan.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 19. Tuliskan komponen minimum bagian Metode Komputasi pada laporan.

Pembahasan. Sebutkan struktur awal dan sumbernya, protonasi, charge, multiplicity, versi perangkat lunak, metode/fungsional, basis per unsur, model pelarut, optimizer, toleransi SCF, grid, kriteria optimasi, frekuensi, temperatur/tekanan, faktor skala, koreksi dispersi/BSSE bila ada, sumber daya, serta prosedur validasi.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

Pertanyaan 20. Rancang uji reproduksibilitas satu job.

Pembahasan. Salin folder job ke lingkungan kedua dengan versi dan dependensi yang dicatat. Jalankan ulang dari input serta konfigurasi yang sama. Bandingkan identitas molekul, jumlah atom, status konvergensi, energi dalam toleransi, geometri RMSD, frekuensi utama, dan checksum file penting. Dokumentasikan perbedaan platform dan nondeterminisme.

Soal pengembangan: Buat contoh molekul atau job kecil yang menguji konsep pada pertanyaan ini. Nyatakan hipotesis, variabel yang diubah, kriteria kelulusan, dan bukti yang harus disimpan.

GLOSARIUM

Glosarium ini mencakup istilah inti aplikasi, kimia kuantum, spektroskopi, cheminformatics, komputasi, dan tata kelola hasil. Huruf A–Z sengaja menggunakan style non-heading sehingga tidak masuk Daftar Isi; hanya judul GLOSARIUM yang terdaftar.

A

Ab initio. Metode struktur elektronik yang dibangun dari prinsip mekanika kuantum tanpa parameterisasi empiris utama untuk sistem tertentu.

Add-on. Perhitungan properti tambahan yang dijalankan setelah atau bersama job utama.

AM1. Hamiltonian semiempirik Austin Model 1.

B

Basis set. Kumpulan fungsi satu-elektron untuk membentuk orbital molekul.

B3LYP. Fungsional hybrid populer yang menggabungkan exchange dan correlation dari beberapa komponen.

BSSE. Basis-set superposition error pada perhitungan interaksi fragmen.

C

Charge. Muatan total sistem yang menentukan jumlah elektron.

Conformer. Struktur dengan konektivitas sama tetapi susunan ruang berbeda.

Cube. Format grid volumetrik Gaussian untuk density, MEP, atau orbital.

D

Density fitting. Aproksimasi integral menggunakan basis bantu untuk mengurangi biaya.

DFT. Density Functional Theory, metode berbasis kerapatan elektron.

Dipole moment. Vektor yang mengukur pemisahan muatan total molekul.

E

Energi elektronik. Energi dari penyelesaian Hamiltonian elektronik pada geometri tertentu.

Embedding. Pembentukan koordinat tiga dimensi dari representasi molekul.

Excited state. Keadaan elektronik di atas keadaan dasar.

F

Faktor skala. Pengali empiris untuk memperbaiki frekuensi harmonik atau ZPE.

Frekuensi imajiner. Mode dengan kelengkungan energi negatif, dilaporkan sebagai nilai negatif.

Fungsional. Aproksimasi exchange–correlation pada DFT.

G

Gap HOMO–LUMO. Selisih energi orbital frontier; deskriptor, bukan otomatis energi serapan.

Geometri. Koordinat atom dan parameter internal molekul.

geomeTRIC. Optimizer geometri berbasis koordinat internal yang terhubung dengan PySCF.

H

Hartree. Satuan energi atomik; sekitar 27,211386 eV.

Hessian. Matriks turunan kedua energi terhadap koordinat.

HOMO. Highest Occupied Molecular Orbital.

I

InChI. International Chemical Identifier berbasis layer struktur.

InChIKey. Hash ringkas dari InChI untuk pencarian.

IR. Spektroskopi inframerah yang berkaitan dengan mode vibrasi aktif dipol.

J

Job. Jenis pekerjaan komputasi: single point, gradien, optimasi, atau frekuensi.

Job worker. Fungsi/thread yang mengeksekusi kumpulan molekul di latar.

JSON. Format teks terstruktur untuk konfigurasi, manifest, dan hasil.

K

Kerapatan elektron. Distribusi probabilitas elektron dalam ruang.

Konvergensi. Keadaan ketika iterasi memenuhi kriteria numerik.

Korelasi elektron. Efek gerak elektron yang tidak tertangkap oleh medan rata-rata sederhana.

L

LUMO. Lowest Unoccupied Molecular Orbital.

Linear dependence. Ketergantungan hampir linear fungsi basis yang mengganggu numerik.

Log. Catatan kronologis proses, status, dan error job.

M

MEP. Molecular electrostatic potential pada ruang sekitar molekul.

Multiplicity. 2S+1, ukuran keadaan spin total.

MP2. Koreksi korelasi Møller–Plesset orde kedua.

Mulliken. Skema partisi populasi berbasis fungsi basis.

N

NBO. Natural Bond Orbital analysis.

NMR. Nuclear Magnetic Resonance; komputasi sering berfokus pada shielding.

Normal mode. Pola gerak vibrasi kolektif hasil diagonalisasi Hessian terbobot massa.

O

Optimizer. Algoritma yang memperbarui koordinat untuk mencari titik stasioner.

ORCA. Paket struktur elektronik yang diakses aplikasi melalui CLI.

Oscillator strength. Ukuran intensitas transisi elektronik.

P

PCM. Polarizable Continuum Model untuk pelarut implisit.

PM7. Hamiltonian semiempirik yang tersedia melalui MOPAC.

Pre-optimasi. Relaksasi awal murah menggunakan force field sebelum QM.

PySCF. Python-based Simulations of Chemistry Framework.

Q

QM. Quantum mechanics; pada konteks buku merujuk komputasi struktur elektronik.

QSAR. Quantitative Structure–Activity Relationship.

QSPR. Quantitative Structure–Property Relationship.

R

Raman. Spektroskopi vibrasi yang terkait perubahan polarizabilitas.

RDKit. Toolkit cheminformatics untuk struktur, identitas, dan deskriptor.

RHF. Restricted Hartree–Fock untuk closed-shell.

ROHF. Restricted open-shell Hartree–Fock.

S

SCF. Self-Consistent Field, iterasi pembentukan kerapatan dan Fock/Kohn–Sham matrix.

Seed. Bilangan awal generator acak untuk reproduksibilitas embedding.

Single point. Perhitungan energi/properti pada geometri tetap.

SMILES. Notasi baris untuk struktur molekul.

T

TDDFT. Time-Dependent DFT untuk eksitasi elektronik.

Termokimia. Perhitungan ZPE, entalpi, entropi, dan energi bebas.

Thread. Unit eksekusi yang dipakai worker latar atau library numerik.

Titik stasioner. Geometri dengan gradien nol dalam toleransi.

U

UFF. Universal Force Field untuk pre-optimasi umum.

UHF. Unrestricted Hartree–Fock dengan orbital alfa dan beta berbeda.

UV–Vis. Spektroskopi serapan ultraviolet–tampak.

V

Valence. Elektron dan orbital terluar yang dominan pada ikatan.

Vektor gradien. Turunan pertama energi terhadap koordinat.

Vibrasi. Gerak periodik atom di sekitar geometri setimbang.

W

Wavefunction. Representasi keadaan kuantum banyak elektron.

Widow/orphan. Baris paragraf tunggal yang terpisah di awal/akhir halaman; dicegah dalam tata letak.

Workflow. Urutan prosedur input, komputasi, validasi, dan pelaporan.

X

XC. Exchange–correlation pada DFT.

XYZ. Format koordinat atom tanpa konektivitas eksplisit.

xTB. Extended tight-binding; keluarga metode GFN untuk komputasi cepat.

Y

Yield. Dalam konteks proses, penyerahan kontrol agar UI tetap responsif; bukan hasil reaksi kimia.

Y-axis. Sumbu ordinat pada plot spektrum, misalnya intensitas atau oscillator strength.

YAML. Format konfigurasi yang tidak dipakai sumber utama tetapi lazim pada workflow reproducible.

Z

Zero-point energy. Energi vibrasi yang tersisa pada 0 K dalam pendekatan kuantum.

Zeta quality. Jumlah fungsi radial untuk merepresentasikan orbital valensi.

ZPE. Singkatan zero-point energy.

DAFTAR PUSTAKA

Daftar pustaka mengutamakan artikel metode, artikel perangkat lunak, buku teks, dan dokumentasi resmi yang mendukung konsep serta workflow dalam buku. Untuk penelitian, pembaca harus mencantumkan versi perangkat lunak yang benar-benar digunakan dan mengikuti rekomendasi sitasi setiap proyek.

[1] Sun, Q., et al. (2018). PySCF: the Python-based simulations of chemistry framework. WIREs Computational Molecular Science, 8, e1340. https://doi.org/10.1002/wcms.1340

[2] Sun, Q., et al. (2020). Recent developments in the PySCF program package. Journal of Chemical Physics, 153, 024109. https://doi.org/10.1063/5.0006074

[3] Smith, D. G. A., et al. (2020). PSI4 1.4: Open-source software for high-throughput quantum chemistry. Journal of Chemical Physics, 152, 184108. https://doi.org/10.1063/5.0006002

[4] RDKit Contributors. RDKit: Open-source cheminformatics. https://www.rdkit.org; citation guidance and version DOIs: https://doi.org/10.5281/zenodo.591637

[5] Weigend, F., & Ahlrichs, R. (2005). Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn. Physical Chemistry Chemical Physics, 7, 3297–3305. https://doi.org/10.1039/B508541A

[6] Dunning, T. H., Jr. (1989). Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. Journal of Chemical Physics, 90, 1007–1023. https://doi.org/10.1063/1.456153

[7] Pritchard, B. P., et al. (2019). A new basis set exchange: An open, up-to-date resource for the molecular sciences community. Journal of Chemical Information and Modeling, 59, 4814–4820. https://doi.org/10.1021/acs.jcim.9b00725

[8] Lehtola, S., Steigemann, C., Oliveira, M. J. T., & Marques, M. A. L. (2018). Recent developments in LIBXC. SoftwareX, 7, 1–5. https://doi.org/10.1016/j.softx.2017.11.002

[9] Hohenberg, P., & Kohn, W. (1964). Inhomogeneous electron gas. Physical Review, 136, B864–B871. https://doi.org/10.1103/PhysRev.136.B864

[10] Kohn, W., & Sham, L. J. (1965). Self-consistent equations including exchange and correlation effects. Physical Review, 140, A1133–A1138. https://doi.org/10.1103/PhysRev.140.A1133

[11] Becke, A. D. (1993). Density-functional thermochemistry. III. The role of exact exchange. Journal of Chemical Physics, 98, 5648–5652. https://doi.org/10.1063/1.464913

[12] Lee, C., Yang, W., & Parr, R. G. (1988). Development of the Colle–Salvetti correlation-energy formula into a functional of the electron density. Physical Review B, 37, 785–789. https://doi.org/10.1103/PhysRevB.37.785

[13] Perdew, J. P., Burke, K., & Ernzerhof, M. (1996). Generalized gradient approximation made simple. Physical Review Letters, 77, 3865–3868. https://doi.org/10.1103/PhysRevLett.77.3865

[14] Adamo, C., & Barone, V. (1999). Toward reliable density functional methods without adjustable parameters: The PBE0 model. Journal of Chemical Physics, 110, 6158–6170. https://doi.org/10.1063/1.478522

[15] Zhao, Y., & Truhlar, D. G. (2008). The M06 suite of density functionals for main group thermochemistry, kinetics, noncovalent interactions, excited states, and transition elements. Theoretical Chemistry Accounts, 120, 215–241. https://doi.org/10.1007/s00214-007-0310-x

[16] Yanai, T., Tew, D. P., & Handy, N. C. (2004). A new hybrid exchange–correlation functional using the Coulomb-attenuating method (CAM-B3LYP). Chemical Physics Letters, 393, 51–57. https://doi.org/10.1016/j.cplett.2004.06.011

[17] Chai, J.-D., & Head-Gordon, M. (2008). Long-range corrected hybrid density functionals with damped atom–atom dispersion corrections. Physical Chemistry Chemical Physics, 10, 6615–6620. https://doi.org/10.1039/B810189B

[18] Møller, C., & Plesset, M. S. (1934). Note on an approximation treatment for many-electron systems. Physical Review, 46, 618–622. https://doi.org/10.1103/PhysRev.46.618

[19] Wang, L.-P., & Song, C. (2016). Geometry optimization made simple with translation and rotation coordinates. Journal of Chemical Physics, 144, 214108. https://doi.org/10.1063/1.4952956

[20] Bannwarth, C., Ehlert, S., & Grimme, S. (2019). GFN2-xTB—An accurate and broadly parametrized self-consistent tight-binding quantum chemical method. Journal of Chemical Theory and Computation, 15, 1652–1671. https://doi.org/10.1021/acs.jctc.8b01176

[21] Grimme, S., Bannwarth, C., & Shushkov, P. (2017). A robust and accurate tight-binding quantum chemical method for structures, vibrational frequencies, and noncovalent interactions. Journal of Chemical Theory and Computation, 13, 1989–2009. https://doi.org/10.1021/acs.jctc.7b00118

[22] Stewart, J. J. P. (2007). Optimization of parameters for semiempirical methods V: Modification of NDDO approximations and application to 70 elements. Journal of Molecular Modeling, 13, 1173–1213. https://doi.org/10.1007/s00894-007-0233-4

[23] Rocha, G. B., et al. (2006). RM1: A reparameterization of AM1 for H, C, N, O, P, S, F, Cl, Br, and I. Journal of Computational Chemistry, 27, 1101–1111. https://doi.org/10.1002/jcc.20425

[24] Neese, F. (2022). Software update: The ORCA program system—Version 5.0. WIREs Computational Molecular Science, 12, e1606. https://doi.org/10.1002/wcms.1606

[25] Tomasi, J., Mennucci, B., & Cammi, R. (2005). Quantum mechanical continuum solvation models. Chemical Reviews, 105, 2999–3094. https://doi.org/10.1021/cr9904009

[26] Cossi, M., Scalmani, G., Rega, N., & Barone, V. (2002). New developments in the polarizable continuum model for quantum mechanical and classical calculations on molecules in solution. Journal of Chemical Physics, 117, 43–54. https://doi.org/10.1063/1.1480445

[27] Casida, M. E. (1995). Time-dependent density functional response theory for molecules. In Recent Advances in Density Functional Methods, Part I. World Scientific. https://doi.org/10.1142/9789812830586_0005

[28] Dreuw, A., & Head-Gordon, M. (2005). Single-reference ab initio methods for the calculation of excited states of large molecules. Chemical Reviews, 105, 4009–4037. https://doi.org/10.1021/cr0505627

[29] Laury, M. L., et al. (2011). Harmonic vibrational frequencies: scale factors for pure, hybrid, hybrid meta, and double-hybrid functionals. Journal of Computational Chemistry, 32, 2339–2347. https://doi.org/10.1002/jcc.21798

[30] Sinha, P., et al. (2004). Harmonic vibrational frequencies: scaling factors for HF, B3LYP, and MP2 with correlation-consistent basis sets. Journal of Physical Chemistry A, 108, 9213–9217. https://doi.org/10.1021/jp048233q

[31] Irikura, K. K., Johnson, R. D., & Kacker, R. N. (2005). Uncertainties in scaling factors for ab initio vibrational frequencies. Journal of Physical Chemistry A, 109, 8430–8437. https://doi.org/10.1021/jp052793n

[32] McQuarrie, D. A. (2008). Quantum Chemistry, 2nd ed. University Science Books.

[33] Szabo, A., & Ostlund, N. S. (1996). Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory. Dover.

[34] Jensen, F. (2017). Introduction to Computational Chemistry, 3rd ed. Wiley.

[35] Cramer, C. J. (2013). Essentials of Computational Chemistry: Theories and Models, 2nd ed. Wiley.

[36] Helgaker, T., Jørgensen, P., & Olsen, J. (2000). Molecular Electronic-Structure Theory. Wiley.

[37] Young, D. C. (2001). Computational Chemistry: A Practical Guide for Applying Techniques to Real-World Problems. Wiley.

[38] PySCF Developers. PySCF User Guide. https://pyscf.org/user/index.html

[39] Psi4 Developers. PSI4 Manual. https://psicode.org/psi4manual/

[40] RDKit Contributors. The RDKit Documentation. https://www.rdkit.org/docs/

[41] Streamlit. Streamlit Documentation. https://docs.streamlit.io/

[42] SciPy Developers. SciPy Optimize Documentation. https://docs.scipy.org/doc/scipy/reference/optimize.html

[43] NumPy Developers. NumPy Documentation. https://numpy.org/doc/

[44] pandas Developers. pandas Documentation. https://pandas.pydata.org/docs/

[45] Grimme Lab. xTB Documentation. https://xtb-docs.readthedocs.io/

[46] ORCA. ORCA 6 Manual. https://www.faccts.de/docs/orca/6.0/manual/

[47] NIST. Computational Chemistry Comparison and Benchmark Database: Vibrational Scaling Factors. https://cccbdb.nist.gov/vibscalejust.asp

LAMPIRAN A — REFERENSI FUNGSI KODE SUMBER

Lampiran ini mendokumentasikan 123 definisi fungsi yang ditemukan melalui analisis sintaks kode sumber. Keterangan peran merupakan pembacaan struktural berdasarkan nama, argumen, docstring, pemanggilan, return, raise, dan cuplikan sumber. Untuk perilaku final, kode sumber tetap menjadi rujukan otoritatif.

Tabel 10. Distribusi fungsi berdasarkan peran yang diinferensikan

Peran	Jumlah
membaca dan menormalkan input struktur molekul	11
membangun representasi, geometri, visualisasi, atau file keluaran	8
mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi	38
mengelola siklus hidup job dan status UI	8
menghitung properti atau add-on khusus	2
mengubah hasil menjadi tabel atau deskriptor	9
menjaga persistensi, serialisasi, dan jejak audit	11
menjalankan atau menyiapkan komputasi PySCF	18
menjalankan workflow ORCA	2
menjalankan workflow xTB	1
menulis, menjalankan, atau mem-parsing workflow MOPAC	15

A.1 ensure_python_package()

Lokasi dan signature: Baris 26–55; ensure_python_package(import_name, pip_name)

Fungsi `ensure_python_package` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 2 pernyataan return dan 2 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Import paket; bila belum ada, instal otomatis dengan interpreter aktif. Nama paket pip hanya berasal dari pemetaan internal kode ini (bukan input user), sehingga instalasi otomatis tetap terkontrol. Untuk executable eksternal seperti ORCA/MOPAC/xTB/Open Babel, instalasi sistem tetap tidak dapat dipaksakan via pip.

Interaksi utama yang terdeteksi: RuntimeError, check_call, import_module, invalidate_caches, join, print, split. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def ensure_python_package(import_name: str, pip_name: str | None = None):
    """Import paket; bila belum ada, instal otomatis dengan interpreter aktif.

    Nama paket pip hanya berasal dari pemetaan internal kode ini (bukan input user),
    sehingga instalasi otomatis tetap terkontrol. Untuk executable eksternal seperti
    ORCA/MOPAC/xTB/Open Babel, instalasi sistem tetap tidak dapat dipaksakan via pip.
    """
    try:
        return importlib.import_module(import_name)
    except ModuleNotFoundError:
        package = pip_name or import_name.split(".", 1)[0]
        cmd = [
            sys.executable, "-m", "pip", "install",
            "--disable-pip-version-check", "--prefer-binary", package,
        ]
        print(f"[AUTO-INSTALL] Dependensi '{import_name}' belum tersedia. Menjalankan: {'
'.join(cmd)}")
        try:
            subprocess.check_call(cmd)
        except Exception as exc:
            raise RuntimeError(
                f"Dependensi Python '{import_name}' belum tersedia dan instalasi otomatis
gagal. "
                f"Perintah yang dicoba: {' '.join(cmd)}. Detail: {exc}"
            ) from exc
        importlib.invalidate_caches()
        try:
            return importlib.import_module(import_name)
```

A.2 safe_slug()

Lokasi dan signature: Baris 302–304; safe_slug(text)

Fungsi `safe_slug` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: str, strip, sub. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def safe_slug(text: str) -> str:
    text = re.sub(r"[^A-Za-z0-9._-]+", "_", str(text).strip())
    return text.strip("._-") or "molecule"
```

A.3 now_text()

Lokasi dan signature: Baris 307–308; now_text()

Fungsi `now_text` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: now, strftime. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def now_text() -> str:
    return datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

A.4 write_json_atomic()

Lokasi dan signature: Baris 311–316; write_json_atomic(path, data)

Fungsi `write_json_atomic` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, dump, open, replace, with_suffix. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def write_json_atomic(path: Path, data):
    path = Path(path)
    tmp = path.with_suffix(path.suffix + ".tmp")
    with open(tmp, "w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=2, allow_nan=True)
    os.replace(tmp, path)
```

A.5 read_json_safe()

Lokasi dan signature: Baris 319–324; read_json_safe(path, default)

Fungsi `read_json_safe` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: load, open. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def read_json_safe(path: Path, default=None):
    try:
        with open(path, "r", encoding="utf-8") as f:
            return json.load(f)
    except Exception:
        return default
```

A.6 append_log()

Lokasi dan signature: Baris 327–331; append_log(path, message)

Fungsi `append_log` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: flush, mkdir, now_text, open, write. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def append_log(path: Path, message: str):
    path.parent.mkdir(parents=True, exist_ok=True)
    with open(path, "a", encoding="utf-8") as f:
        f.write(f"[{now_text()}] {message}\n")
        f.flush()
```

A.7 is_wsl_environment()

> **Lokasi dan signature:** Baris 334–341; is_wsl_environment()

Fungsi `is_wsl_environment` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Deteksi WSL tanpa menambah dependensi eksternal.

Interaksi utama yang terdeteksi: Path, get, lower, read_text. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def is_wsl_environment():
    """Deteksi WSL tanpa menambah dependensi
eksternal."""
    try:
        if os.environ.get("WSL_DISTRO_NAME") or
os.environ.get("WSL_INTEROP"):
            return True
        return "microsoft" in
Path("/proc/version").read_text(encoding="utf-8",
errors="ignore").lower()
    except Exception:
        return False
```

A.8 open_folder_os()

Lokasi dan signature: Baris 344–371; open_folder_os(folder)

Fungsi `open_folder_os` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 6 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Buka folder pada OS host. Khusus WSL gunakan explorer.exe + wslpath -w.

Interaksi utama yang terdeteksi: Path, Popen, expanduser, is_wsl_environment, lower, resolve, run, startfile, startswith, str, strip, system, which. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def open_folder_os(folder: str):
    """Buka folder pada OS host. Khusus WSL gunakan
explorer.exe + wslpath -w."""
    resolved = Path(folder).expanduser().resolve()
    p = str(resolved)
    try:
        system = platform.system().lower()
        if is_wsl_environment():
            cp = subprocess.run(
                ["wslpath", "-w", p],
                stdout=subprocess.PIPE,
stderr=subprocess.PIPE, text=True, timeout=10,
            )
            win_path = cp.stdout.strip() if cp.returncode == 0
else p
            explorer = shutil.which("explorer.exe") or
"/mnt/c/Windows/explorer.exe"
            subprocess.Popen([explorer, win_path],
stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)
            return True, win_path
        if system.startswith("windows"):
            os.startfile(p)  # type: ignore[attr-defined]
            return True, p
        if system == "darwin":
            subprocess.Popen(["open", p])
            return True, p
        opener = shutil.which("xdg-open")
        if opener:
            subprocess.Popen([opener, p],
stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)
            return True, p
        return False, "xdg-open tidak ditemukan"
```

A.9 find_mopac_executable()

Lokasi dan signature: Baris 374–380; find_mopac_executable()

Fungsi `find_mopac_executable` berperan untuk menulis, menjalankan, atau memparsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Cari executable MOPAC yang umum di Linux/WSL/Windows PATH.

Interaksi utama yang terdeteksi: which. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def find_mopac_executable():
    """Cari executable MOPAC yang umum di
Linux/WSL/Windows PATH."""
    for name in ("mopac", "MOPAC", "mopac.exe",
"MOPAC2016.exe", "MOPAC2023.exe", "MOPAC2024.exe"):
        found = shutil.which(name)
        if found:
            return found
    return None
```

A.10 detect_gpu_info()

Lokasi dan signature: Baris 383–409; detect_gpu_info()

Fungsi `detect_gpu_info` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Deteksi CUDA/NVIDIA secara ringan; tidak memaksa PyTorch/CuPy terpasang.

Interaksi utama yang terdeteksi: get, join, len, run, splitlines, strip, update, which. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def detect_gpu_info():
    """Deteksi CUDA/NVIDIA secara ringan; tidak memaksa
PyTorch/CuPy terpasang."""
    info = {
        "detected": False,
        "name": "",
        "count": 0,
        "cuda_visible_devices":
os.environ.get("CUDA_VISIBLE_DEVICES", ""),
        "gpu4pyscf": False,
    }
    nvidia_smi = shutil.which("nvidia-smi")
    if nvidia_smi:
        try:
            cp = subprocess.run(
                [nvidia_smi, "--query-gpu=name", "--
format=csv,noheader"],
                stdout=subprocess.PIPE,
stderr=subprocess.PIPE, text=True, timeout=10,
            )
            names = [x.strip() for x in
cp.stdout.splitlines() if x.strip()]
            if cp.returncode == 0 and names:
                info.update(detected=True, name=",
".join(names), count=len(names))
        except Exception:
            pass
    try:
        import gpu4pyscf  # noqa: F401
        info["gpu4pyscf"] = True
    except Exception:
        pass
```

A.11 energy_units()

Lokasi dan signature: Baris 412–430; energy_units(e_h)

Fungsi `energy_units` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: float. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def energy_units(e_h):
    if e_h is None:
        return {
            "hartree": None,
            "au": None,
            "ev": None,
            "kj_mol": None,
            "kcal_mol": None,
            "cm-1": None,
        }
    e = float(e_h)
    return {
        "hartree": e,
        "au": e,
        "ev": e * HARTREE_TO_EV,
        "kj_mol": e * HARTREE_TO_KJ_MOL,
        "kcal_mol": e * HARTREE_TO_KCAL_MOL,
        "cm-1": e * HARTREE_TO_CM1,
    }
```

A.12 backend_status()

Lokasi dan signature: Baris 433–450; backend_status()

Fungsi `backend_status` berperan untuk mengelola siklus hidup job dan status UI. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: bool, detect_gpu_info, find_mopac_executable, which. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def backend_status():
    semi_ok = False
    try:
        from pyscf import semiempirical  # noqa: F401
        semi_ok = True
    except Exception:
        pass
    gpu = detect_gpu_info()
    return {
        "PySCF": True,
        "PySCF Semiempirical": semi_ok,
        "MOPAC CLI": find_mopac_executable() is not None,
        "xTB CLI": shutil.which("xtb") is not None,
        "ORCA CLI": shutil.which("orca") is not None,
        "Open Babel": shutil.which("obabel") is not None,
        "GPU/CUDA": bool(gpu["detected"]),
        "GPU4PySCF": bool(gpu["gpu4pyscf"]),
    }
```

A.13 build_rdkit_3d()

Lokasi dan signature: Baris 456–461; build_rdkit_3d(smiles, random_seed)

Fungsi `build_rdkit_3d` berperan untuk membangun representasi, geometri, visualisasi, atau file keluaran. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Fungsi lama dipertahankan: bangun struktur 3D dari SMILES.

Interaksi utama yang terdeteksi: MolFromSmiles, ValueError, prepare_molecule_3d. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def build_rdkit_3d(smiles: str, random_seed: int = 20260820):
    """Fungsi lama dipertahankan: bangun struktur 3D dari SMILES."""
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        raise ValueError(f"SMILES tidak valid: {smiles}")
    return prepare_molecule_3d(mol, random_seed=random_seed, preopt=True)
```

A.14 prepare_molecule_3d()

Lokasi dan signature: Baris 464–502; prepare_molecule_3d(mol, random_seed, preopt)

Fungsi `prepare_molecule_3d` berperan untuk membangun representasi, geometri, visualisasi, atau file keluaran. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 2 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: AddHs, ETKDGv3, EmbedMolecule, GetConformer, GetNumConformers, Is3D, MMFFHasAllMoleculeParams, MMFFOptimizeMolecule, Mol, RuntimeError, SanitizeMol, UFFOptimizeMolecule, ValueError, bool, int. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def prepare_molecule_3d(mol, random_seed=20260820,
preopt=True):
    if mol is None:
        raise ValueError("Objek molekul kosong/tidak
valid.")

    mol = Chem.Mol(mol)
    try:
        Chem.SanitizeMol(mol)
    except Exception:
        # Beberapa struktur logam/koordinasi dapat gagal
sanitasi; tetap dicoba.
        pass

    mol = Chem.AddHs(mol, addCoords=True)
    need_embed = mol.GetNumConformers() == 0
    if not need_embed:
```

```
        try:
            need_embed = not
bool(mol.GetConformer().Is3D())
        except Exception:
            need_embed = True

    if need_embed:
        params = AllChem.ETKDGv3()
        params.randomSeed = int(random_seed)
        params.useRandomCoords = True
        status = AllChem.EmbedMolecule(mol, params)
        if status != 0:
            params.useRandomCoords = True
```

A.15 infer_bonds_xyz()

Lokasi dan signature: Baris 505–515; infer_bonds_xyz(mol, charge)

Fungsi `infer_bonds_xyz` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: DetermineBonds, DetermineConnectivity, int. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def infer_bonds_xyz(mol, charge=0):
    if mol is None or rdDetermineBonds is None:
        return mol
    try:
        rdDetermineBonds.DetermineBonds(mol,
charge=int(charge))
    except Exception:
        try:
            rdDetermineBonds.DetermineConnectivity(mol)
        except Exception:
            pass
    return mol
```

A.16 mol_from_openbabel_bytes()

Lokasi dan signature: Baris 518–543; mol_from_openbabel_bytes(filename, data)

Fungsi `mol_from_openbabel_bytes` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 3 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, SDMolSupplier, TemporaryDirectory, ValueError, exists, next, run, str, strip, which, write_bytes. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def mol_from_openbabel_bytes(filename: str, data: bytes):
    obabel = shutil.which("obabel")
    if not obabel:
        raise ValueError(
            f"Format {Path(filename).suffix or '(tanpa
ekstensi)'} belum didukung langsung oleh RDKit. "
            "Pasang Open Babel agar format tambahan dapat
dikonversi otomatis."
        )
    suffix = Path(filename).suffix or ".dat"
    with tempfile.TemporaryDirectory() as td:
        inp = Path(td) / ("input" + suffix)
        out = Path(td) / "converted.sdf"
        inp.write_bytes(data)
        cp = subprocess.run(
            [obabel, str(inp), "-O", str(out)],
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            text=True,
            timeout=120,
        )
        if cp.returncode != 0 or not out.exists():
            raise ValueError(f"Open Babel gagal membaca
{filename}: {cp.stderr.strip()}")
        supplier = Chem.SDMolSupplier(str(out),
removeHs=False)
        mol = next((m for m in supplier if m is not
None), None)
        if mol is None:
            raise ValueError(f"Open Babel menghasilkan
SDF, tetapi RDKit gagal membacanya: {filename}")
        return mol
```

A.17 molecule_from_bytes()

> **Lokasi dan signature:** Baris 546–583; molecule_from_bytes(filename, data, charge_hint)

Fungsi `molecule_from_bytes` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: BytesIO, ForwardSDMolSupplier, MolFromInchi, MolFromMol2Block, MolFromMolBlock, MolFromPDBBlock, MolFromSmiles, MolFromXYZBlock, Path, decode, infer_bonds_xyz, lower, mol_from_openbabel_bytes, next, split, splitlines, startswith, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
    - Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def molecule_from_bytes(filename: str, data: bytes,
charge_hint=0):
    ext = Path(filename).suffix.lower()
    text = None

    if ext in {".sdf", ".sd"}:
        supplier =
Chem.ForwardSDMolSupplier(io.BytesIO(data), removeHs=False)
        mol = next((m for m in supplier if m is not None),
None)
    elif ext in {".mol", ".mdl"}:
        text = data.decode("utf-8", errors="replace")
        mol = Chem.MolFromMolBlock(text, removeHs=False,
sanitize=True)
    elif ext == ".mol2":
        text = data.decode("utf-8", errors="replace")
        mol = Chem.MolFromMol2Block(text, removeHs=False,
sanitize=True)
    elif ext in {".pdb", ".ent"}:
        text = data.decode("utf-8", errors="replace")
        mol = Chem.MolFromPDBBlock(text, removeHs=False,
sanitize=True)
    elif ext in {".xyz"}:
        text = data.decode("utf-8", errors="replace")
        mol = Chem.MolFromXYZBlock(text)
        mol = infer_bonds_xyz(mol, charge=charge_hint)
    elif ext in {".smi", ".smiles", ".can"}:
        text = data.decode("utf-8", errors="replace").strip()
        token = text.splitlines()[0].strip().split()[0]
        mol = Chem.MolFromSmiles(token)
    elif ext in {".inchi"}:
        text = data.decode("utf-8",
errors="replace").strip().splitlines()[0]
```

A.18 molecule_from_local_path()

Lokasi dan signature: Baris 586–590; molecule_from_local_path(path_text, charge_hint)

Fungsi `molecule_from_local_path` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: FileNotFoundError, Path, exists, expanduser, is_file, molecule_from_bytes, read_bytes. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def molecule_from_local_path(path_text: str, charge_hint=0):
    path = Path(path_text).expanduser()
    if not path.exists() or not path.is_file():
        raise FileNotFoundError(f"File tidak ditemukan: {path}")
    return molecule_from_bytes(path.name, path.read_bytes(), charge_hint=charge_hint)
```

A.19 molecule_from_identifier()

Lokasi dan signature: Baris 593–603; molecule_from_identifier(identifier)

Fungsi `molecule_from_identifier` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 2 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: MolFromInchi, MolFromSmiles, ValueError, startswith, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def molecule_from_identifier(identifier: str):
    s = identifier.strip()
    if not s:
        raise ValueError("SMILES/InChI kosong.")
    if s.startswith("InChI="):
        mol = Chem.MolFromInchi(s)
    else:
        mol = Chem.MolFromSmiles(s)
    if mol is None:
        raise ValueError(f"SMILES/InChI tidak valid:
{s}")
    return mol
```

A.20 molecule_identifiers()

Lokasi dan signature: Baris 606–634; molecule_identifiers(mol)

Fungsi `molecule_identifiers` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: CalcExactMolWt, CalcMolFormula, Mol, MolToInchi, MolToInchiKey, MolToSmiles, MolWt, RemoveHs, float. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def molecule_identifiers(mol):
    heavy = Chem.RemoveHs(Chem.Mol(mol))
    formula = rdMolDescriptors.CalcMolFormula(heavy)
    try:
        canonical = Chem.MolToSmiles(heavy,
canonical=True, isomericSmiles=True)
    except Exception:
        canonical = ""
    try:
        inchi = Chem.MolToInchi(heavy)
    except Exception:
        inchi = ""
    try:
        inchikey = Chem.MolToInchiKey(heavy)
    except Exception:
        inchikey = ""
    try:
        mw = float(Descriptors.MolWt(heavy))
        exact =
float(rdMolDescriptors.CalcExactMolWt(heavy))
```

```
    except Exception:
        mw = float("nan")
        exact = float("nan")
    return {
        "formula": formula,
        "smiles": canonical,
        "inchi": inchi,
        "inchikey": inchikey,
```

A.21 formal_charge()

Lokasi dan signature: Baris 637–641; formal_charge(mol)

Fungsi `formal_charge` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetAtoms, GetFormalCharge, int, sum. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def formal_charge(mol):
    try:
        return int(sum(a.GetFormalCharge() for a in mol.GetAtoms()))
    except Exception:
        return 0
```

A.22 mol_to_pyscf_atoms()

Lokasi dan signature: Baris 644–650; mol_to_pyscf_atoms(rdkit_mol)

Fungsi `mol_to_pyscf_atoms` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetAtomPosition, GetAtoms, GetConformer, GetSymbol, append, enumerate, float. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def mol_to_pyscf_atoms(rdkit_mol):
    conf = rdkit_mol.GetConformer()
    atoms = []
    for i, atom in enumerate(rdkit_mol.GetAtoms()):
        p = conf.GetAtomPosition(i)
        atoms.append([atom.GetSymbol(), (float(p.x),
float(p.y), float(p.z))])
    return atoms
```

A.23 rdkit_with_new_coords()

Lokasi dan signature: Baris 653–665; rdkit_with_new_coords(template_mol, coords_angstrom)

Fungsi `rdkit_with_new_coords` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: AddConformer, Conformer, GetConformer, GetNumAtoms, GetNumConformers, Mol, Set3D, SetAtomPosition, ValueError, asarray, enumerate, float, len. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def rdkit_with_new_coords(template_mol, coords_angstrom):
    mol = Chem.Mol(template_mol)
    if mol.GetNumConformers() == 0:
        conf = Chem.Conformer(mol.GetNumAtoms())
        mol.AddConformer(conf, assignId=True)
    conf = mol.GetConformer()
    conf.Set3D(True)
    coords = np.asarray(coords_angstrom, dtype=float)
    if len(coords) != mol.GetNumAtoms():
        raise ValueError("Jumlah koordinat final tidak
sama dengan jumlah atom pada struktur awal.")
```

```
    for i, (x, y, z) in enumerate(coords):
        conf.SetAtomPosition(i, (float(x), float(y),
float(z)))
    return mol
```

A.24 mol_to_xyz()

Lokasi dan signature: Baris 668–674; mol_to_xyz(mol, title)

Fungsi `mol_to_xyz` berperan untuk membangun representasi, geometri, visualisasi, atau file keluaran. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetAtomPosition, GetAtoms, GetConformer, GetNumAtoms, GetSymbol, append, enumerate, join, str. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def mol_to_xyz(mol, title="molecule"):
    conf = mol.GetConformer()
    lines = [str(mol.GetNumAtoms()), title]
    for i, atom in enumerate(mol.GetAtoms()):
        p = conf.GetAtomPosition(i)
        lines.append(f"{atom.GetSymbol():2s} {p.x: .10f}
{p.y: .10f} {p.z: .10f}")
    return "\n".join(lines) + "\n"
```

A.25 export_bonded_formats()

Lokasi dan signature: Baris 677–689; export_bonded_formats(mol, title)

Fungsi `export_bonded_formats` berperan untuk membangun representasi, geometri, visualisasi, atau file keluaran. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: ClearProp, HasProp, Mol, MolToMolBlock, MolToPDBBlock, SetProp, mol_to_xyz. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def export_bonded_formats(mol, title="optimized molecule"):
    mol = Chem.Mol(mol)
    if mol.HasProp("_Name"):
        mol.ClearProp("_Name")
    mol.SetProp("_Name", title)
    molblock = Chem.MolToMolBlock(mol)
    sdf = molblock + "\n$$$$\n"
    try:
        pdb = Chem.MolToPDBBlock(mol)
    except Exception:
        pdb = ""
    xyz = mol_to_xyz(mol, title=title)
    return {"mol": molblock + "\n", "sdf": sdf, "pdb": pdb,
"xyz": xyz}
```

A.26 _as_json_value()

Lokasi dan signature: Baris 694–709; _as_json_value(value)

Fungsi `_as_json_value` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 7 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Konversi scalar/array NumPy menjadi objek JSON yang aman.

Interaksi utama yang terdeteksi: _as_json_value, float, isinstance, item, str, tolist. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _as_json_value(value):
    """Konversi scalar/array NumPy menjadi objek JSON
yang aman."""
    if value is None:
        return None
    if isinstance(value, (str, bool, int, float)):
        return value
    if isinstance(value, np.generic):
        return value.item()
    if isinstance(value, np.ndarray):
        return value.tolist()
    if isinstance(value, (tuple, list)):
        return [_as_json_value(v) for v in value]
    try:
        return float(value)
    except Exception:
        return str(value)
```

A.27 calculate_geometry_tables()

Lokasi dan signature: Baris 712–793; calculate_geometry_tables(rdmol)

Fungsi `calculate_geometry_tables` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Semua panjang ikatan, sudut ikatan, dan dihedral dari geometri final.

Interaksi utama yang terdeteksi: GetAngleDeg, GetAtomPosition, GetAtomWithIdx, GetBeginAtomIdx, GetBondLength, GetBondTypeAsDouble, GetBonds, GetConformer, GetDihedralDeg, GetEndAtomIdx, GetIdx, GetIsAromatic, GetNeighbors, GetNumAtoms, GetSymbol, Mol, add, append. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def calculate_geometry_tables(rdmol):
    """Semua panjang ikatan, sudut ikatan, dan dihedral
dari geometri final."""
    from rdkit.Chem import rdMolTransforms

    mol = Chem.Mol(rdmol)
    conf = mol.GetConformer()
    bonds = []
    for bond in mol.GetBonds():
        i = bond.GetBeginAtomIdx()
        j = bond.GetEndAtomIdx()
        try:
            length =
float(rdMolTransforms.GetBondLength(conf, i, j))
        except Exception:
```

```
            pi = conf.GetAtomPosition(i)
            pj = conf.GetAtomPosition(j)
            length = float(math.sqrt((pi.x-pj.x)**2 +
(pi.y-pj.y)**2 + (pi.z-pj.z)**2))
        bonds.append({
            "Bond": len(bonds) + 1,
            "Atom 1": i + 1,
            "Unsur 1": mol.GetAtomWithIdx(i).GetSymbol(),
            "Atom 2": j + 1,
            "Unsur 2": mol.GetAtomWithIdx(j).GetSymbol(),
            "Panjang (Å)": length,
            "Orde ikatan":
float(bond.GetBondTypeAsDouble()),
            "Aromatik": bool(bond.GetIsAromatic()),
        })
```

A.28 calculate_rdkit_qsar_qspr_descriptors()

> **Lokasi dan signature:** Baris 796–865;
> calculate_rdkit_qsar_qspr_descriptors(rdmol)

Fungsi `calculate_rdkit_qsar_qspr_descriptors` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Koleksi deskriptor RDKit 2D + 3D untuk QSAR/QSPR secara luas.

Interaksi utama yang terdeteksi: ComputeGasteigerCharges, ComputeMolVolume, GetAtoms, GetNumConformers, GetProp, Mol, RemoveHs, SanitizeMol, abs, asarray, float, fn, getattr, isfinite, isscalar, items, lower, max. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def calculate_rdkit_qsar_qspr_descriptors(rdmol):
    """Koleksi deskriptor RDKit 2D + 3D untuk QSAR/QSPR
secara luas."""
    mol3d = Chem.Mol(rdmol)
    try:
        Chem.SanitizeMol(mol3d)
    except Exception:
        pass
    try:
        mol2d = Chem.RemoveHs(Chem.Mol(mol3d))
    except Exception:
        mol2d = Chem.Mol(mol3d)

    values = {}
    # Seluruh scalar descriptor bawaan RDKit (umumnya
>200 descriptor pada rilis modern).
    for name, fn in getattr(Descriptors, "descList", []):
        try:
            v = fn(mol2d)
            if np.isscalar(v):
                fv = float(v)
                if math.isfinite(fv):
                    values[name] = (fv, "RDKit 2D")
        except Exception:
            pass

    # Deskriptor 3D penting untuk QSPR/QSAR konformasi.
    three_d = {
```

A.29 add_qm_qsar_descriptors()

Lokasi dan signature: Baris 868–916; add_qm_qsar_descriptors(rows, result)

Fungsi `add_qm_qsar_descriptors` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Tambahkan descriptor konseptual-DFT/QM ke tabel QSAR/QSPR.

Interaksi utama yang terdeteksi: abs, add, append, float, get, isfinite, list. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def add_qm_qsar_descriptors(rows, result):
    """Tambahkan descriptor konseptual-DFT/QM ke tabel
QSAR/QSPR."""
    out = list(rows or [])
    def add(name, value, category="QM / conceptual DFT"):
        if value is None:
            return
        try:
            fv = float(value)
            if math.isfinite(fv):
                out.append({"Descriptor": name, "Value":
fv, "Category": category})
        except Exception:
            pass

    add("QM_TotalEnergy_Hartree",
result.get("energy_final_h"), "QM")
    add("QM_HOMO_eV", result.get("homo_ev"), "QM")
    add("QM_LUMO_eV", result.get("lumo_ev"), "QM")
    add("QM_HOMO_LUMO_Gap_eV", result.get("gap_ev"),
"QM")
    add("QM_Dipole_X_D", result.get("dipole_x_d"), "QM")
    add("QM_Dipole_Y_D", result.get("dipole_y_d"), "QM")
    add("QM_Dipole_Z_D", result.get("dipole_z_d"), "QM")
    add("QM_Dipole_Magnitude_D",
result.get("dipole_total_d"), "QM")

    h = result.get("homo_ev")
    l = result.get("lumo_ev")
    if h is not None and l is not None:
        try:
```

A.30 add()

Lokasi dan signature: Baris 871–879; add(name, value, category)

Fungsi `add` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: append, float, isfinite. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def add(name, value, category="QM / conceptual DFT"):
        if value is None:
            return
        try:
            fv = float(value)
            if math.isfinite(fv):
                out.append({"Descriptor": name, "Value":
fv, "Category": category})
        except Exception:
            pass
```

A.31 build_optimization_geometry_energy_rows()

Lokasi dan signature: Baris 919–960;
build_optimization_geometry_energy_rows(result)

Fungsi `build_optimization_geometry_energy_rows` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: append, get. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def build_optimization_geometry_energy_rows(result):
    rows = []
    for r in result.get("bond_lengths", []):
        rows.append({
            "Jenis": "Bond", "Parameter": f"{r.get('Atom
1')}-{r.get('Atom 2')}",
            "Nilai": r.get("Panjang (Å)"), "Satuan": "Å",
            "Detail": f"{r.get('Unsur 1')}-{r.get('Unsur
2')}; order={r.get('Orde ikatan')};
aromatic={r.get('Aromatik')}",
        })
    for r in result.get("bond_angles", []):
        rows.append({
            "Jenis": "Angle", "Parameter": f"{r.get('Atom
1')}-{r.get('Atom 2 (pusat)')}-{r.get('Atom 3')}",
            "Nilai": r.get("Sudut (deg)"), "Satuan":
"deg",
            "Detail": f"{r.get('Unsur 1')}-{r.get('Unsur
2')}-{r.get('Unsur 3')}",
        })
    for r in result.get("dihedral_angles", []):
        rows.append({
            "Jenis": "Dihedral", "Parameter":
f"{r.get('Atom 1')}-{r.get('Atom 2')}-{r.get('Atom 3')}-
{r.get('Atom 4')}",
            "Nilai": r.get("Dihedral (deg)"), "Satuan":
"deg",
            "Detail": f"{r.get('Unsur 1')}-{r.get('Unsur
2')}-{r.get('Unsur 3')}-{r.get('Unsur 4')}",
        })
```

```
    e = result.get("energy") or {}
    for key, unit in (("hartree", "Eh"), ("ev", "eV"),
("kj_mol", "kJ/mol"), ("kcal_mol", "kcal/mol"), ("cm-1",
"cm-1")):
        if e.get(key) is not None:
            rows.append({"Jenis": "Energy", "Parameter":
f"Total energy ({key})", "Nilai": e.get(key), "Satuan":
unit, "Detail": "Final electronic energy"})
    de = result.get("delta_energy") or {}
    for key, unit in (("hartree", "Eh"), ("ev", "eV"),
("kj_mol", "kJ/mol"), ("kcal_mol", "kcal/mol")):
```

A.32 enrich_result_common()

Lokasi dan signature: Baris 963–984; enrich_result_common(result)

Fungsi `enrich_result_common` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Pengayaan non-destruktif untuk semua backend: geometri + QSAR/QSPR.

Interaksi utama yang terdeteksi: MolFromMolBlock, add_qm_qsar_descriptors, append, build_optimization_geometry_energy_rows, calculate_geometry_tables, calculate_rdkit_qsar_qspr_descriptors, get, setdefault. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def enrich_result_common(result):
    """Pengayaan non-destruktif untuk semua backend: geometri + QSAR/QSPR."""
    result.setdefault("addon_meta", {"name": result.get("addon", "None"), "status":
"backend_not_implemented" if result.get("addon", "None") != "None" else "not_requested"})
```

```
    result.setdefault("addon_results", [])
    result.setdefault("thermochemistry", [])
    result.setdefault("cube_files", {})
    result.setdefault("mep_stats", {})
    try:
        mol = Chem.MolFromMolBlock(result.get("mol", ""), removeHs=False, sanitize=False)
        if mol is None:
            return result
        bonds, angles, dihedrals = calculate_geometry_tables(mol)
        result["bond_lengths"] = bonds
        result["bond_angles"] = angles
        result["dihedral_angles"] = dihedrals
        result["qsar_qspr_descriptors"] = add_qm_qsar_descriptors(
            calculate_rdkit_qsar_qspr_descriptors(mol), result
        )
        result["optimization_geometry_energies"] =
build_optimization_geometry_energy_rows(result)
    except Exception as exc:
        result.setdefault("analysis_warnings", []).append(f"Analisis geometri/QSAR gagal:
{exc}")
    return result
```

A.33 _spin_summed_dm()

Lokasi dan signature: Baris 987–994; _spin_summed_dm(mf)

Fungsi `_spin_summed_dm` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: asarray, isinstance, make_rdm1. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.

- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _spin_summed_dm(mf):
    dm = mf.make_rdm1()
    if isinstance(dm, (tuple, list)):
        return np.asarray(dm[0]) + np.asarray(dm[1])
    arr = np.asarray(dm)
    if arr.ndim == 3 and arr.shape[0] == 2:
        return arr[0] + arr[1]
    return arr
```

A.34 _frontier_mo_coefficients()

Lokasi dan signature: Baris 997–1017; _frontier_mo_coefficients(mf)

Fungsi `_frontier_mo_coefficients` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: append, asarray, float, int, isinstance, list, max, min, where, zip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _frontier_mo_coefficients(mf):
    e = mf.mo_energy
    occ = mf.mo_occ
    coeff = mf.mo_coeff
    channels = []
    if isinstance(e, (tuple, list)) or (isinstance(e, np.ndarray) and np.asarray(e).ndim
== 2):
        labels = ["alpha", "beta"]
        for label, en, oc, co in zip(labels, list(e), list(occ), list(coeff)):
            channels.append((label, np.asarray(en, dtype=float), np.asarray(oc,
dtype=float), np.asarray(co)))
    else:
        channels.append(("R", np.asarray(e, dtype=float), np.asarray(occ, dtype=float),
np.asarray(coeff)))
    occ_cand = []
    vir_cand = []
    for label, en, oc, co in channels:
        for idx in np.where(oc > 1e-10)[0]:
            occ_cand.append((float(en[idx]), label, int(idx), co[:, idx]))
        for idx in np.where(oc <= 1e-10)[0]:
            vir_cand.append((float(en[idx]), label, int(idx), co[:, idx]))
    homo = max(occ_cand, key=lambda x: x[0]) if occ_cand else None
    lumo = min(vir_cand, key=lambda x: x[0]) if vir_cand else None
    return homo, lumo
```

A.35 generate_pyscf_cube_files()

Lokasi dan signature: Baris 1020–1065; generate_pyscf_cube_files(mf, mol, mol_dir, base, grid_points, progress_log)

Fungsi `generate_pyscf_cube_files` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 6; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: MEP, density, HOMO dan LUMO Gaussian cube dari wavefunction final PySCF.

Interaksi utama yang terdeteksi: Path, _frontier_mo_coefficients, _spin_summed_dm, append_log, asarray, density, float, int, isfinite, max, mean, mep, min, nanmax, nanmin, orbital, str. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def generate_pyscf_cube_files(mf, mol, mol_dir, base, grid_points=60, progress_log=None):
    """MEP, density, HOMO dan LUMO Gaussian cube dari wavefunction final PySCF."""
    from pyscf.tools import cubegen
    mol_dir = Path(mol_dir)
    n = int(grid_points)
    dm = _spin_summed_dm(mf)
    files = {}
    stats = {}
    density_path = mol_dir / f"{base}_density.cube"
    mep_path = mol_dir / f"{base}_mep.cube"
    homo_path = mol_dir / f"{base}_homo.cube"
    lumo_path = mol_dir / f"{base}_lumo.cube"

    if progress_log:
        append_log(Path(progress_log), f"Membuat cube grid {n}x{n}x{n}:
density/MEP/HOMO/LUMO.")
    try:
        dens = np.asarray(cubegen.density(mol, str(density_path), dm, nx=n, ny=n, nz=n),
dtype=float)
        files["Electron density cube"] = str(density_path)
        stats["density_min"] = float(np.nanmin(dens)); stats["density_max"] =
float(np.nanmax(dens))
    except Exception as exc:
        if progress_log: append_log(Path(progress_log), f"Density cube gagal: {exc}")
    try:
        mep = np.asarray(cubegen.mep(mol, str(mep_path), dm, nx=n, ny=n, nz=n),
dtype=float)
        files["MEP cube"] = str(mep_path)
        finite = mep[np.isfinite(mep)]
        if finite.size:
```

A.36 run_pyscf_uvvis()

> **Lokasi dan signature:** Baris 1068–1102; run_pyscf_uvvis(mf, nroots, progress_log)

Fungsi `run_pyscf_uvvis` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: TDDFT/TDHF/TDA excited states + oscillator strengths untuk UV-Vis.

Interaksi utama yang terdeteksi: Path, RuntimeError, append, append_log, asarray, astype, enumerate, factory, float, full_like, getattr, int, isfinite, kernel, len, oscillator_strength, real. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_pyscf_uvvis(mf, nroots=20, progress_log=None):
    """TDDFT/TDHF/TDA excited states + oscillator strengths untuk UV-Vis."""
    if progress_log: append_log(Path(progress_log), f"Add-On UV-Vis: menghitung
{int(nroots)} excited states.")
    td = None
    method_used = ""
    # DFT objects biasanya memiliki TDDFT; HF memiliki TDHF. TDA menjadi fallback robust.
    for attr, label in (("TDDFT", "TDDFT"), ("TDHF", "TDHF"), ("TDA", "TDA")):
        try:
            factory = getattr(mf, attr)
            td = factory()
            method_used = label
            break
        except Exception:
            continue
    if td is None:
        raise RuntimeError("Objek mean-field ini tidak menyediakan TDDFT/TDHF/TDA.")
    td.nstates = int(nroots)
    td.kernel()
    energies_h = np.real(np.asarray(td.e, dtype=complex)).astype(float)
    try:
        osc = np.real(np.asarray(td.oscillator_strength(), dtype=complex)).astype(float)
    except Exception:
        osc = np.full_like(energies_h, np.nan, dtype=float)
    rows = []
    for i, eh in enumerate(energies_h, start=1):
        ev = float(eh * HARTREE_TO_EV)
```

A.37 run_pyscf_thermochemistry()

Lokasi dan signature: Baris 1105–1126; run_pyscf_thermochemistry(mf, mol, hessian, temperature, pressure, progress_log)

Fungsi `run_pyscf_thermochemistry` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 6; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, RuntimeError, _as_json_value, append, append_log, asarray, float, harmonic_analysis, iscomplexobj, isfinite, isinstance, isreal, items, len, real, thermo. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_pyscf_thermochemistry(mf, mol, hessian, temperature=DEFAULT_TEMPERATURE_K,
pressure=DEFAULT_PRESSURE_PA, progress_log=None):
    from pyscf.hessian import thermo as pyscf_thermo
    if progress_log: append_log(Path(progress_log), f"Add-On Thermodynamics:
T={temperature} K, P={pressure} Pa.")
    harmonic = pyscf_thermo.harmonic_analysis(mol, np.asarray(hessian),
imaginary_freq=False)
    freq_au = np.asarray(harmonic["freq_au"])
    # Thermochemistry ideal-gas/harmonic hanya memakai mode vibrasi nyata positif.
    # Mode imajiner/zero mode tetap tersedia di tabel frekuensi utama, tetapi tidak
    # dimasukkan ke fungsi partisi agar log/eksponensial tidak menghasilkan NaN palsu.
    freq_real = np.real(freq_au[np.isreal(freq_au)]) if np.iscomplexobj(freq_au) else
np.asarray(freq_au, dtype=float)
```

```
    freq_positive = np.asarray(freq_real, dtype=float)
    freq_positive = freq_positive[np.isfinite(freq_positive) & (freq_positive > 1.0e-12)]
    if freq_positive.size == 0:
        raise RuntimeError("Tidak ada frekuensi vibrasi positif yang dapat dipakai untuk
thermochemistry.")
    tdata = pyscf_thermo.thermo(mf, freq_positive, float(temperature), float(pressure))
    rows = []
    for key, value in tdata.items():
        if isinstance(value, tuple) and len(value) == 2:
            val, unit = value
        else:
            val, unit = value, ""
        rows.append({"Property": key, "Value": _as_json_value(val), "Unit": unit})
    return rows
```

A.38 run_pyscf_nmr()

Lokasi dan signature: Baris 1129–1169; run_pyscf_nmr(mf, progress_log)

Fungsi `run_pyscf_nmr` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 2 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: NMR shielding melalui PySCF properties extension bila tersedia/kompatibel.

Interaksi utama yang terdeteksi: Path, RuntimeError, append, append_log, asarray, atom_symbol, cls, eigvalsh, ensure_python_package, float, getattr, kernel, range, trace, upper. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_pyscf_nmr(mf, progress_log=None):
```

```
    """NMR shielding melalui PySCF properties extension bila tersedia/kompatibel."""
    if progress_log: append_log(Path(progress_log), "Add-On NMR: memuat pyscf properties
extension.")
    try:
        ensure_python_package("pyscf.prop.nmr", "pyscf[properties]")
        from pyscf.prop import nmr
    except Exception as exc:
        raise RuntimeError(f"PySCF properties/NMR tidak tersedia: {exc}") from exc

    cls_name = mf.__class__.__name__.upper()
    candidates = []
    if "UKS" in cls_name: candidates = ["UKS", "UHF"]
    elif "RKS" in cls_name: candidates = ["RKS", "RHF"]
    elif "UHF" in cls_name: candidates = ["UHF"]
    else: candidates = ["RHF"]
    obj = None
    for name in candidates:
        cls = getattr(nmr, name, None)
        if cls is not None:
            obj = cls(mf)
            break
    if obj is None:
        raise RuntimeError(f"Driver NMR tidak ditemukan untuk {cls_name}.")
    shielding = np.asarray(obj.kernel(), dtype=float)
    rows = []
    for ia in range(shielding.shape[0]):
```

A.39 _write_xyz_body()

Lokasi dan signature: Baris 1172–1178; _write_xyz_body(rdmol)

Fungsi `_write_xyz_body` berperan untuk membangun representasi, geometri, visualisasi, atau file keluaran. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetAtomPosition, GetAtoms, GetConformer, GetSymbol, append, enumerate. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _write_xyz_body(rdmol):
    conf = rdmol.GetConformer()
    lines = []
    for i, atom in enumerate(rdmol.GetAtoms()):
        p = conf.GetAtomPosition(i)
        lines.append(f"{atom.GetSymbol():2s} {p.x: .10f} {p.y: .10f} {p.z: .10f}")
    return lines
```

A.40 run_orca_raman_addon()

> **Lokasi dan signature:** Baris 1181–1216; run_orca_raman_addon(rdmol, config, spec, mol_dir, stop_event, progress_log)

Fungsi `run_orca_raman_addon` berperan untuk menjalankan workflow ORCA. Jumlah argumen posisi yang terdeteksi adalah 6; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Raman ORCA 6.1+: Freq + analytical polarizability derivative untuk HF/DFT.

Interaksi utama yang terdeteksi: Path, _write_xyz_body, append, exists, finditer, float, get, group, int, join, len, max, read_text, run_process_interruptible, str, which, write_text. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.

- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_orca_raman_addon(rdmol, config, spec, mol_dir, stop_event, progress_log):
    """Raman ORCA 6.1+: Freq + analytical polarizability derivative untuk HF/DFT."""
    mol_dir = Path(mol_dir)
    inp = mol_dir / "addon_raman_orca.inp"
    method = config.get("method", "B3LYP")
    # Raman adapter memakai ORCA. Bila nama metode PySCF tidak tersedia pada
    # daftar ORCA aplikasi, gunakan B3LYP alih-alih menghasilkan input invalid.
    if method not in ORCA_METHODS:
        method = "B3LYP"
    basis = config.get("basis", "def2-SVP")
    charge = int(spec["charge"]); mult = int(spec["multiplicity"])
    nprocs = int(config.get("threads", 1)); maxcore = max(100,
int(config.get("max_memory_mb", 4000) // max(1, nprocs)))
    lines = [
        f"! {method} {basis} Freq TightSCF",
        f"%pal nprocs {nprocs} end",
        f"%maxcore {maxcore}",
        "%elprop Polar 1 end",
        f"* xyz {charge} {mult}",
        *_write_xyz_body(rdmol),
        "*",
    ]
    inp.write_text("\n".join(lines) + "\n", encoding="utf-8")
    orca = shutil.which("orca")
    if not orca:
        return {"status": "input_generated", "message": "ORCA tidak ditemukan; input Raman
sudah dibuat.", "input_file": str(inp)}, []
    outlog = mol_dir / "addon_raman_orca.out"
```

A.41 run_g3mp2_addon()

> **Lokasi dan signature:** Baris 1219–1260; run_g3mp2_addon(rdmol, spec, mol_dir, stop_event, progress_log)

Fungsi `run_g3mp2_addon` berperan untuk menghitung properti atau add-on khusus. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 2 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Hook G3MP2: jalankan Gaussian bila ada; bila tidak, simpan input tanpa memalsukan ΔHf.

Interaksi utama yang terdeteksi: Path, Popen, StopRequested, _write_xyz_body, findall, float, int, is_set, items, join, next, open, poll, read_text, replace, sleep, str, terminate. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_g3mp2_addon(rdmol, spec, mol_dir, stop_event, progress_log):
    """Hook G3MP2: jalankan Gaussian bila ada; bila tidak, simpan input tanpa memalsukan
ΔHf."""
    mol_dir = Path(mol_dir)
    gjf = mol_dir / "addon_g3mp2.gjf"
    charge = int(spec["charge"]); mult = int(spec["multiplicity"])
    lines = [
        "%chk=addon_g3mp2.chk",
        "#p G3MP2",
        "",
        "G3MP2 add-on generated by QM KOMPUTASI V2.3.4",
        "",
        f"{charge} {mult}",
        *_write_xyz_body(rdmol),
        "",
    ]
    gjf.write_text("\n".join(lines) + "\n", encoding="utf-8")
    exe = next((shutil.which(x) for x in ("g16", "g09", "gaussian") if shutil.which(x)),
None)
    if not exe:
        return {"status": "input_generated", "message": "Gaussian G3MP2 executable tidak
ditemukan; input .gjf sudah dibuat. ΔHf tidak direkayasa.", "input_file": str(gjf)}
    outlog = mol_dir / "addon_g3mp2.out"
    # Gaussian umum menerima input melalui stdin.
    with open(gjf, "rb") as fin, open(outlog, "wb") as fout:
        proc = subprocess.Popen([exe], cwd=str(mol_dir), stdin=fin, stdout=fout,
stderr=subprocess.STDOUT)
        while proc.poll() is None:
            if stop_event.is_set():
                proc.terminate()
```

A.42 run_nbo_addon()

Lokasi dan signature: Baris 1263–1291; run_nbo_addon(rdmol, spec, config, mol_dir, stop_event, progress_log)

Fungsi `run_nbo_addon` berperan untuk menghitung properti atau add-on khusus. Jumlah argumen posisi yang terdeteksi adalah 6; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: NBO def2-TZPP melalui ORCA + standalone NBO6/7 jika NBOEXE tersedia.

Interaksi utama yang terdeteksi: Path, _write_xyz_body, append, exists, get, int, join, max, run_process_interruptible, str, strip, which, write_text. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_nbo_addon(rdmol, spec, config, mol_dir, stop_event, progress_log):
    """NBO def2-TZPP melalui ORCA + standalone NBO6/7 jika NBOEXE tersedia."""
    mol_dir = Path(mol_dir)
    inp = mol_dir / "addon_nbo_def2-TZPP.inp"
    method = config.get("method", "B3LYP")
    # Jika metode utama bukan HF/DFT yang lazim di ORCA, gunakan B3LYP sebagai adapter NBO
yang eksplisit.
    if method not in ORCA_METHODS:
        method = "B3LYP"
    charge = int(spec["charge"]); mult = int(spec["multiplicity"])
    nprocs = int(config.get("threads", 1)); maxcore = max(100,
int(config.get("max_memory_mb", 4000) // max(1, nprocs)))
    lines = [
        f"! {method} def2-TZPP SP TightSCF NBO",
        f"%pal nprocs {nprocs} end",
        f"%maxcore {maxcore}",
        f"* xyz {charge} {mult}",
        *_write_xyz_body(rdmol),
        "*",
    ]
    inp.write_text("\n".join(lines) + "\n", encoding="utf-8")
    orca = shutil.which("orca")
    nboexe = os.environ.get("NBOEXE", "").strip()
    if not orca or not nboexe or not Path(nboexe).exists():
        missing = []
```

```
        if not orca: missing.append("ORCA")
        if not nboexe or not Path(nboexe).exists(): missing.append("NBOEXE/NBO6-7")
        return {"status": "input_generated", "message": f"Input NBO dibuat; belum
dijalankan karena {', '.join(missing)} tidak tersedia.", "input_file": str(inp), "basis":
"def2-TZPP"}
```

A.43 apply_selected_addon_pyscf()

Lokasi dan signature: Baris 1294–1338; apply_selected_addon_pyscf(addon, mf, mol, rdmol, spec, config, mol_dir, stop_event, progress_log, hessian)

Fungsi `apply_selected_addon_pyscf` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 10; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Hessian, Path, _pyscf_hessian_to_frequencies, append_log, asarray, float, get, int, kernel, len, run_g3mp2_addon, run_nbo_addon, run_orca_raman_addon, run_pyscf_nmr, run_pyscf_thermochemistry, run_pyscf_uvvis, str, update. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def apply_selected_addon_pyscf(addon, mf, mol, rdmol, spec, config, mol_dir, stop_event,
progress_log, hessian=None):
```

```
    addon = addon or "None"
    meta = {"name": addon, "status": "not_requested" if addon == "None" else "started"}
    rows = []
    thermo_rows = []
    if addon == "None":
        return meta, rows, thermo_rows, hessian
    try:
        if addon == "IR":
            if hessian is None:
                hessian = np.asarray(mf.Hessian().kernel(), dtype=float)
            rows = _pyscf_hessian_to_frequencies(mol, hessian)
            meta.update(
                status="completed",
                message=f"IR harmonik selesai; {len(rows)} mode vibrasi dihitung tanpa
thermochemistry.",
            )
        elif addon == "Thermodynamics":
            if hessian is None:
                hessian = np.asarray(mf.Hessian().kernel(), dtype=float)
            thermo_rows = run_pyscf_thermochemistry(
                mf, mol, hessian,
                temperature=float(config.get("temperature_k", DEFAULT_TEMPERATURE_K)),
                pressure=float(config.get("pressure_pa", DEFAULT_PRESSURE_PA)),
                progress_log=progress_log,
            )
            meta.update(status="completed", message="Thermodynamics harmonik selesai.")
```

A.44 mol_3d_html()

Lokasi dan signature: Baris 1340–1351; mol_3d_html(mol, width, height)

Fungsi `mol_3d_html` berperan untuk membangun representasi, geometri, visualisasi, atau file keluaran. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: MolToMolBlock, _make_html, addModel, ensure_python_package, setBackgroundColor, setStyle, view, zoomTo. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def mol_3d_html(mol, width=700, height=430):
    try:
        ensure_python_package("py3Dmol", "py3Dmol")
        import py3Dmol
        view = py3Dmol.view(width=width, height=height)
        view.addModel(Chem.MolToMolBlock(mol), "mol")
        view.setStyle({"stick": {}, "sphere": {"scale": 0.28}})
        view.zoomTo()
        view.setBackgroundColor("white")
        return view._make_html()
    except Exception:
        return None
```

A.45 pyscf_xyz()

Lokasi dan signature: Baris 1357–1364; pyscf_xyz(mol, title)

Fungsi `pyscf_xyz` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: append, atom_coords, atom_symbol, join, range, str. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.

- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def pyscf_xyz(mol, title: str) -> str:
    coords = mol.atom_coords(unit="Angstrom")
    lines = [str(mol.natm), title]
    for i in range(mol.natm):
        sym = mol.atom_symbol(i)
        x, y, z = coords[i]
        lines.append(f"{sym:2s} {x: .10f} {y: .10f} {z: .10f}")
    return "\n".join(lines) + "\n"
```

A.46 coordinates_dataframe()

Lokasi dan signature: Baris 1367–1378; coordinates_dataframe(mol)

Fungsi `coordinates_dataframe` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: DataFrame, append, atom_coords, atom_symbol, float, range. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def coordinates_dataframe(mol):
    coords = mol.atom_coords(unit="Angstrom")
    rows = []
    for i in range(mol.natm):
        rows.append({
            "Atom": i + 1,
            "Unsur": mol.atom_symbol(i),
            "X (Å)": float(coords[i, 0]),
            "Y (Å)": float(coords[i, 1]),
            "Z (Å)": float(coords[i, 2]),
        })
    return pd.DataFrame(rows)
```

A.47 orbital_dataframe()

Lokasi dan signature: Baris 1381–1399; orbital_dataframe(mf)

Fungsi `orbital_dataframe` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: DataFrame, append, asarray, enumerate, float, isinstance, list, zip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def orbital_dataframe(mf):
    e = mf.mo_energy
    occ = mf.mo_occ
    rows = []
    if isinstance(e, (tuple, list)) or (isinstance(e, np.ndarray) and np.asarray(e).ndim
== 2):
        energies = list(e)
        occs = list(occ)
        labels = ["α", "β"]
        for spin_label, en, oc in zip(labels, energies, occs):
            en = np.asarray(en, dtype=float) * HARTREE_TO_EV
            oc = np.asarray(oc, dtype=float)
            for i, (ee, oo) in enumerate(zip(en, oc), start=1):
                rows.append({"Spin": spin_label, "MO": i, "Occupancy": float(oo), "Energi
(eV)": float(ee)})
    else:
        en = np.asarray(e, dtype=float) * HARTREE_TO_EV
        oc = np.asarray(occ, dtype=float)
        for i, (ee, oo) in enumerate(zip(en, oc), start=1):
            rows.append({"Spin": "R", "MO": i, "Occupancy": float(oo), "Energi (eV)":
float(ee)})
    return pd.DataFrame(rows)
```

A.48 calculate_frontier_orbitals()

Lokasi dan signature: Baris 1402–1438; calculate_frontier_orbitals(mf)

Fungsi `calculate_frontier_orbitals` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: append, asarray, float, int, isinstance, list, max, min, where, zip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def calculate_frontier_orbitals(mf):
    e = mf.mo_energy
    occ = mf.mo_occ
    candidates_occ = []
    candidates_vir = []

    if isinstance(e, (tuple, list)) or (isinstance(e, np.ndarray) and np.asarray(e).ndim
== 2):
        for label, en, oc in zip(["α", "β"], list(e), list(occ)):
            en = np.asarray(en, dtype=float)
            oc = np.asarray(oc, dtype=float)
            for idx in np.where(oc > 0)[0]:
                candidates_occ.append((float(en[idx]), label, int(idx + 1)))
            for idx in np.where(oc == 0)[0]:
                candidates_vir.append((float(en[idx]), label, int(idx + 1)))
    else:
        en = np.asarray(e, dtype=float)
        oc = np.asarray(occ, dtype=float)
        for idx in np.where(oc > 0)[0]:
            candidates_occ.append((float(en[idx]), "R", int(idx + 1)))
        for idx in np.where(oc == 0)[0]:
            candidates_vir.append((float(en[idx]), "R", int(idx + 1)))

    if not candidates_occ or not candidates_vir:
        return {
            "homo_index": None, "lumo_index": None,
            "homo_ev": None, "lumo_ev": None, "gap_ev": None,
```

A.49 mulliken_dataframe()

Lokasi dan signature: Baris 1441–1454; mulliken_dataframe(mf)

Fungsi `mulliken_dataframe` berperan untuk mengubah hasil menjadi tabel atau deskriptor. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: DataFrame, append, asarray, atom_symbol, float, mulliken_pop, range. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def mulliken_dataframe(mf):
    try:
        _, charges = mf.mulliken_pop(verbose=0)
        charges = np.asarray(charges, dtype=float)
        rows = []
        for i in range(mf.mol.natm):
            rows.append({
                "Atom": i + 1,
                "Unsur": mf.mol.atom_symbol(i),
                "Muatan Mulliken (e)": float(charges[i]),
            })
        return pd.DataFrame(rows)
    except Exception:
        return pd.DataFrame(columns=["Atom", "Unsur", "Muatan Mulliken (e)"])
```

A.50 make_pyscf_molecule()

Lokasi dan signature: Baris 1457–1470; make_pyscf_molecule(atoms, basis, charge, spin, max_memory_mb, log_handle)

Fungsi `make_pyscf_molecule` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 6; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: M, dict, int. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def make_pyscf_molecule(atoms, basis="def2-SVP", charge=0, spin=0, max_memory_mb=4000,
log_handle=None):
    kwargs = dict(
        atom=atoms,
        unit="Angstrom",
        basis=basis,
        charge=int(charge),
        spin=int(spin),
        verbose=4 if log_handle else 0,
        max_memory=int(max_memory_mb),
    )
    mol = gto.M(**kwargs)
    if log_handle is not None:
        mol.stdout = log_handle
    return mol
```

A.51 make_pyscf_method()

Lokasi dan signature: Baris 1473–1535; make_pyscf_method(mol, method_name, grid_level, conv_tol, max_cycle, density_fitting, chkfile, log_handle, stop_event, progress_log, use_gpu, gpu_threads)

Fungsi `make_pyscf_method` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 12; tubuh fungsi memiliki 1 pernyataan return dan 3 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, RHF, RKS, ROHF, StopRequested, UHF, UKS, ValueError, append_log, density_fit, float, get, hasattr, int, is_set, str, to_gpu. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def make_pyscf_method(mol, method_name, grid_level=3, conv_tol=1e-9, max_cycle=120,
                      density_fitting=False, chkfile=None, log_handle=None,
stop_event=None,
                      progress_log=None, use_gpu=False, gpu_threads=1):
    meta = PYSCF_METHODS[method_name]
    family = meta["family"]

    if family == "HF" or family == "MP2":
        ref = meta.get("hf_reference", "AUTO")
        if family == "MP2":
            ref = "AUTO"
        if ref == "RHF":
            if mol.spin != 0:
```

```
                raise ValueError("RHF memerlukan closed-shell: spin=0 / multiplicitas 1.
Gunakan ROHF atau UHF untuk open-shell.")
            mf = scf.RHF(mol)
        elif ref == "ROHF":
            mf = scf.ROHF(mol)
        elif ref == "UHF":
            mf = scf.UHF(mol)
        else:
            mf = scf.UHF(mol) if mol.spin != 0 else scf.RHF(mol)
    elif family == "DFT":
        mf = dft.UKS(mol) if mol.spin != 0 else dft.RKS(mol)
        mf.xc = meta["xc"]
        mf.grids.level = int(grid_level)
    else:
        raise ValueError(f"Family metode tidak dikenal: {family}")
```

A.52 scf_callback()

Lokasi dan signature: Baris 1526–1532; scf_callback(envs)

Fungsi `scf_callback` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 0 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: StopRequested, append_log, get, is_set. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def scf_callback(envs):
        if stop_event is not None and stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
        cyc = envs.get("cycle", envs.get("cycle_id", "?"))
        e_tot = envs.get("e_tot", envs.get("last_hf_e", None))
        if progress_log and e_tot is not None:
            append_log(progress_log, f"SCF cycle {cyc}: E = {e_tot}")
```

A.53 make_b3lyp()

> **Lokasi dan signature:** Baris 1538–1543; make_b3lyp(mol, grid_level, conv_tol, max_cycle)

Fungsi `make_b3lyp` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 4; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Fungsi lama dipertahankan untuk kompatibilitas internal.

Interaksi utama yang terdeteksi: make_pyscf_method. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def make_b3lyp(mol, grid_level=3, conv_tol=1e-9, max_cycle=100):
    """Fungsi lama dipertahankan untuk kompatibilitas internal."""
    return make_pyscf_method(
        mol, "B3LYP", grid_level=grid_level,
        conv_tol=conv_tol, max_cycle=max_cycle,
    )
```

A.54 run_scf_with_fallback()

Lokasi dan signature: Baris 1546–1565; run_scf_with_fallback(mf, conv_tol, max_cycle, stop_event, progress_log)

Fungsi `run_scf_with_fallback` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 2 pernyataan return dan 2 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: RuntimeError, StopRequested, append_log, float, getattr, int, is_set, kernel, newton. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_scf_with_fallback(mf, conv_tol, max_cycle, stop_event, progress_log):
    if stop_event.is_set():
        raise StopRequested("Perhitungan dihentikan oleh pengguna.")
    e = float(mf.kernel())
    if getattr(mf, "converged", False):
        return mf, e

    append_log(progress_log, "SCF biasa belum konvergen; mencoba second-order/Newton
SCF.")
    newton_mf = mf.newton()
    newton_mf.conv_tol = float(conv_tol)
    newton_mf.max_cycle = int(max_cycle)
```

```
newton_mf.callback = mf.callback
try:
    newton_mf.stdout = mf.stdout
except Exception:
    pass
e = float(newton_mf.kernel())
if not getattr(newton_mf, "converged", False):
    raise RuntimeError("SCF tidak konvergen, termasuk setelah fallback Newton.")
return newton_mf, e
```

A.55 optimize_pyscf()

Lokasi dan signature: Baris 1568–1627; optimize_pyscf(mf, optimizer_name, maxsteps, stop_event, progress_log)

Fungsi `optimize_pyscf` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 4 pernyataan return dan 3 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: StopRequested, ValueError, append, append_log, attach_history, berny_opt, ensure_python_package, float, geometric_opt, get, getattr, int, is_set, kernel, list, nuc_grad_method, optimizer, setattr. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def optimize_pyscf(mf, optimizer_name, maxsteps, stop_event, progress_log):
    if stop_event.is_set():
        raise StopRequested("Perhitungan dihentikan oleh pengguna.")

    counter = {"n": 0, "energies": []}

    def opt_callback(envs):
        counter["n"] += 1
        if stop_event.is_set():
            raise StopRequested("Optimasi dihentikan oleh pengguna.")
        e = None
        try:
            scanner = envs.get("g_scanner")
            e = getattr(scanner, "e_tot", None) if scanner is not None else None
        except Exception:
            e = None
        if e is None:
            e = envs.get("energy", envs.get("e_tot"))
        try:
            e_num = float(e) if e is not None else None
        except Exception:
            e_num = None
        counter["energies"].append({
            "Step": int(counter["n"]),
            "Energy (Eh)": e_num,
            "Energy (eV)": (e_num * HARTREE_TO_EV if e_num is not None else None),
```

A.56 opt_callback()

Lokasi dan signature: Baris 1574–1597; opt_callback(envs)

Fungsi `opt_callback` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 0 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: StopRequested, append, append_log, float, get, getattr, int, is_set. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def opt_callback(envs):
        counter["n"] += 1
        if stop_event.is_set():
            raise StopRequested("Optimasi dihentikan oleh pengguna.")
        e = None
        try:
            scanner = envs.get("g_scanner")
            e = getattr(scanner, "e_tot", None) if scanner is not None else None
        except Exception:
            e = None
        if e is None:
            e = envs.get("energy", envs.get("e_tot"))
        try:
            e_num = float(e) if e is not None else None
        except Exception:
            e_num = None
        counter["energies"].append({
            "Step": int(counter["n"]),
            "Energy (Eh)": e_num,
            "Energy (eV)": (e_num * HARTREE_TO_EV if e_num is not None else None),
            "Energy (kJ/mol)": (e_num * HARTREE_TO_KJ_MOL if e_num is not None else None),
            "Energy (kcal/mol)": (e_num * HARTREE_TO_KCAL_MOL if e_num is not None else
None),
        })
        append_log(progress_log, f"Geometry step {counter['n']}: E = {e}")
```

A.57 attach_history()

Lokasi dan signature: Baris 1599–1604; attach_history(mol_out)

Fungsi `attach_history` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: list, setattr. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def attach_history(mol_out):
        try:
            setattr(mol_out, "_qm_opt_energy_history", list(counter["energies"]))
        except Exception:
            pass
        return mol_out
```

A.58 _patch_semiempirical_init_guess()

Lokasi dan signature: Baris 1630–1673;
_patch_semiempirical_init_guess(mf, progress_log)

Fungsi `_patch_semiempirical_init_guess` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 5 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Compatibility shim for pyscf-semiempirical vs newer PySCF SCF API. Newer PySCF kernels call ``mf.get_init_guess(..., s1e=s1e)``. Older pyscf-semiempirical classes (RAM1/UAM1/RMINDO3/UMINDO3) expose ``get_init_guess(mol=None, key='minao')`` without ``s1e``/``**kwargs``. Patch only the *instance* so the external package itself is never modified.

Interaksi utama yang terdeteksi: MethodType, Path, any, append_log, original_get_init_guess, signature, values. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _patch_semiempirical_init_guess(mf, progress_log=None):
    """Compatibility shim for pyscf-semiempirical vs newer PySCF SCF API.

    Newer PySCF kernels call ``mf.get_init_guess(..., s1e=s1e)``.  Older
    pyscf-semiempirical classes (RAM1/UAM1/RMINDO3/UMINDO3) expose
    ``get_init_guess(mol=None, key='minao')`` without ``s1e``/``**kwargs``.
    Patch only the *instance* so the external package itself is never modified.
    """
    try:
        sig = inspect.signature(mf.get_init_guess)
        accepts_s1e = "s1e" in sig.parameters
        accepts_kwargs = any(
            p.kind == inspect.Parameter.VAR_KEYWORD
            for p in sig.parameters.values()
        )
        if accepts_s1e or accepts_kwargs:
            return mf
    except Exception:
        # If signature inspection fails, install the harmless compatibility
        # wrapper below.  It delegates to the original bound method.
        pass

    original_get_init_guess = mf.get_init_guess

    def get_init_guess_compat(self, mol=None, key="minao", s1e=None, **kwargs):
        # ``s1e`` is intentionally ignored.  Semiempirical classes construct
```

A.59 get_init_guess_compat()

Lokasi dan signature: Baris 1654–1665; get_init_guess_compat(self, mol, key, s1e)

Fungsi `get_init_guess_compat` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 4; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: original_get_init_guess. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def get_init_guess_compat(self, mol=None, key="minao", s1e=None, **kwargs):
        # ``s1e`` is intentionally ignored.  Semiempirical classes construct
        # their own minimal/orthogonalized working basis and their historical
        # initial-guess implementation does not need the overlap passed by the
        # generic PySCF SCF kernel.
        try:
            return original_get_init_guess(mol, key)
        except TypeError:
            try:
                return original_get_init_guess(mol)
            except TypeError:
                return original_get_init_guess()
```

A.60 _configure_semiempirical_mf()

Lokasi dan signature: Baris 1676–1698; _configure_semiempirical_mf(cls, mol, config, log_handle, stop_event, progress_log, verbose)

Fungsi `_configure_semiempirical_mf` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 7; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Create and configure a semiempirical SCF object consistently.

Interaksi utama yang terdeteksi: Path, StopRequested, _patch_semiempirical_init_guess, append_log, cls, float, get, int, is_set. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _configure_semiempirical_mf(cls, mol, config, log_handle=None,
                                stop_event=None, progress_log=None,
                                verbose=True):
    """Create and configure a semiempirical SCF object consistently."""
    mf = cls(mol)
    _patch_semiempirical_init_guess(mf, progress_log=progress_log)
    mf.conv_tol = float(config["scf_conv_tol"])
    mf.max_cycle = int(config["max_scf_cycle"])
    mf.verbose = 4 if (log_handle is not None and verbose) else 0
    if log_handle is not None:
        mf.stdout = log_handle

    def semi_scf_callback(envs):
        if stop_event is not None and stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
        if progress_log is not None:
            cyc = envs.get("cycle", envs.get("cycle_id", None))
            e_tot = envs.get("e_tot", envs.get("last_hf_e", None))
            if cyc is not None and e_tot is not None:
                append_log(Path(progress_log), f"Semiempirical SCF cycle {cyc}: E =
{e_tot}")

    mf.callback = semi_scf_callback
    return mf
```

A.61 semi_scf_callback()

Lokasi dan signature: Baris 1688–1695; semi_scf_callback(envs)

Fungsi `semi_scf_callback` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 0 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, StopRequested, append_log, get, is_set. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def semi_scf_callback(envs):
        if stop_event is not None and stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
        if progress_log is not None:
            cyc = envs.get("cycle", envs.get("cycle_id", None))
            e_tot = envs.get("e_tot", envs.get("last_hf_e", None))
            if cyc is not None and e_tot is not None:
                append_log(Path(progress_log), f"Semiempirical SCF cycle {cyc}: E =
{e_tot}")
```

A.62 _semiempirical_energy_for_coords()

Lokasi dan signature: Baris 1701–1730;
_semiempirical_energy_for_coords(symbols, coords_angstrom, cls, charge, spin, config, stop_event, progress_log, log_handle, verbose)

Fungsi `_semiempirical_energy_for_coords` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 10; tubuh fungsi memiliki 1 pernyataan return dan 2 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Evaluate one AM1/MINDO3 energy at Cartesian coordinates in Angstrom.

Interaksi utama yang terdeteksi: M, RuntimeError, StopRequested, _configure_semiempirical_mf, asarray, float, getattr, int, is_set, kernel, reshape, zip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _semiempirical_energy_for_coords(symbols, coords_angstrom, cls, charge, spin,
                                     config, stop_event, progress_log=None,
                                     log_handle=None, verbose=False):
    """Evaluate one AM1/MINDO3 energy at Cartesian coordinates in Angstrom."""
    if stop_event is not None and stop_event.is_set():
        raise StopRequested("Perhitungan dihentikan oleh pengguna.")

    atoms = [
        [sym, (float(x), float(y), float(z))]
        for sym, (x, y, z) in zip(symbols, np.asarray(coords_angstrom).reshape(-1, 3))
    ]
    mol = gto.M(
        atom=atoms,
        unit="Angstrom",
        charge=int(charge),
        spin=int(spin),
```

```
        verbose=0,
        max_memory=int(config["max_memory_mb"]),
    )
    mf = _configure_semiempirical_mf(
        cls, mol, config,
        log_handle=log_handle,
        stop_event=stop_event,
        progress_log=progress_log if verbose else None,
        verbose=verbose,
    )
```

A.63 _numerical_cartesian_gradient()

Lokasi dan signature: Baris 1733–1767;
_numerical_cartesian_gradient(symbols, coords_angstrom, cls, charge, spin, config, stop_event, progress_log, base_energy, step_angstrom)

Fungsi `_numerical_cartesian_gradient` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 10; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Forward finite-difference gradient in Eh/Angstrom. AM1 closed-shell in pyscf-semiempirical currently lacks an analytical nuclear-gradient implementation. A finite-difference fallback keeps AM1 usable for gradient/optimization jobs without changing the Hamiltonian.

Interaksi utama yang terdeteksi: Path, StopRequested, _semiempirical_energy_for_coords, abs, append_log, asarray, copy, float, is_set, max, mean, range, reshape, sqrt, zeros_like. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _numerical_cartesian_gradient(symbols, coords_angstrom, cls, charge, spin,
                                  config, stop_event, progress_log=None,
                                  base_energy=None, step_angstrom=1.0e-3):
    """Forward finite-difference gradient in Eh/Angstrom.

    AM1 closed-shell in pyscf-semiempirical currently lacks an analytical
    nuclear-gradient implementation.  A finite-difference fallback keeps AM1
    usable for gradient/optimization jobs without changing the Hamiltonian.
    """
    x = np.asarray(coords_angstrom, dtype=float).reshape(-1)
    if base_energy is None:
        base_energy, _, _ = _semiempirical_energy_for_coords(
            symbols, x.reshape(-1, 3), cls, charge, spin, config,
            stop_event, progress_log=None, verbose=False,
        )
    grad = np.zeros_like(x)
    h = float(step_angstrom)
    for idx in range(x.size):
        if stop_event is not None and stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
        xp = x.copy()
        xp[idx] += h
        ep, _, _ = _semiempirical_energy_for_coords(
            symbols, xp.reshape(-1, 3), cls, charge, spin, config,
            stop_event, progress_log=None, verbose=False,
        )
```

A.64 _optimize_semiempirical_numerical()

> **Lokasi dan signature:** Baris 1770–1868; _optimize_semiempirical_numerical(symbols, coords0_angstrom, cls, charge, spin, config, stop_event, progress_log)

Fungsi `_optimize_semiempirical_numerical` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 8; tubuh fungsi memiliki 6 pernyataan return dan 5 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Fallback Cartesian L-BFGS-B optimizer using semiempirical energies. Used when the selected semiempirical method does not expose a working analytical ``nuc_grad_method``. The electronic method remains AM1/MINDO3; only the geometry gradient is evaluated numerically.

Interaksi utama yang terdeteksi: Path, RuntimeError, StopRequested, _numerical_cartesian_gradient, _semiempirical_energy_for_coords, abs, append_log, array_equal, asarray, copy, ensure_python_package, float, getattr, int, is_set, jacobian, max, mean. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _optimize_semiempirical_numerical(symbols, coords0_angstrom, cls, charge, spin,
                                      config, stop_event, progress_log):
    """Fallback Cartesian L-BFGS-B optimizer using semiempirical energies.

    Used when the selected semiempirical method does not expose a working
    analytical ``nuc_grad_method``.  The electronic method remains AM1/MINDO3;
    only the geometry gradient is evaluated numerically.
    """
    try:
        ensure_python_package("scipy", "scipy")
        from scipy.optimize import minimize
    except Exception as exc:
        raise RuntimeError(
            f"Optimasi numerik AM1 memerlukan SciPy dan instalasi otomatis gagal: {exc}"
        ) from exc

    progress_path = Path(progress_log)
    x0 = np.asarray(coords0_angstrom, dtype=float).reshape(-1)
    cache = {"x": None, "e": None, "grad": None, "eval": 0, "iter": 0}

    def same_x(x):
        return cache["x"] is not None and np.array_equal(np.asarray(x), cache["x"])

    def objective(x):
        if stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
```

A.65 same_x()

Lokasi dan signature: Baris 1790–1791; same_x(x)

Fungsi `same_x` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: array_equal, asarray. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def same_x(x):
        return cache["x"] is not None and np.array_equal(np.asarray(x), cache["x"])
```

A.66 objective()

Lokasi dan signature: Baris 1793–1808; objective(x)

Fungsi `objective` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 2 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: StopRequested, _semiempirical_energy_for_coords, append_log, asarray, copy, float, is_set, reshape, same_x. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def objective(x):
        if stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
        x = np.asarray(x, dtype=float)
        if same_x(x) and cache["e"] is not None:
            return float(cache["e"])
        e, _, _ = _semiempirical_energy_for_coords(
            symbols, x.reshape(-1, 3), cls, charge, spin, config,
            stop_event, progress_log=None, verbose=False,
        )
        cache["x"] = x.copy()
        cache["e"] = float(e)
        cache["grad"] = None
        cache["eval"] += 1
        append_log(progress_path, f"{config['method']} numerical optimizer energy eval
{cache['eval']}: E = {e:.12f} Eh")
        return float(e)
```

A.67 jacobian()

Lokasi dan signature: Baris 1810–1825; jacobian(x)

Fungsi `jacobian` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 2 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: StopRequested, _numerical_cartesian_gradient, asarray, copy, is_set, objective, reshape, same_x. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def jacobian(x):
        if stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
        x = np.asarray(x, dtype=float)
        e0 = objective(x)
        if same_x(x) and cache["grad"] is not None:
            return cache["grad"].copy()
        g = _numerical_cartesian_gradient(
            symbols, x.reshape(-1, 3), cls, charge, spin, config,
            stop_event, progress_log=progress_path,
            base_energy=e0, step_angstrom=1.0e-3,
        ).reshape(-1)
        cache["x"] = x.copy()
        cache["e"] = e0
        cache["grad"] = g.copy()
        return g
```

A.68 opt_callback()

Lokasi dan signature: Baris 1827–1838; opt_callback(xk)

Fungsi `opt_callback` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 0 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: StopRequested, abs, append_log, is_set, jacobian, max, mean, objective, sqrt. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def opt_callback(xk):
        cache["iter"] += 1
        e = objective(xk)
        g = jacobian(xk)
        append_log(
            progress_path,
            f"{config['method']} numerical optimization step {cache['iter']}: E={e:.12f}
Eh; "
            f"RMSgrad={np.sqrt(np.mean(g**2)):.6e} Eh/Angstrom; "
            f"MAXgrad={np.max(np.abs(g)):.6e} Eh/Angstrom",
        )
        if stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
```

A.69 run_pyscf_semiempirical()

Lokasi dan signature: Baris 1871–2050; run_pyscf_semiempirical(spec, config, mol_dir, stop_event, progress_log)

Fungsi `run_pyscf_semiempirical` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 1 pernyataan return dan 5 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: M, MolFromMolBlock, Path, RuntimeError, _configure_semiempirical_mf, _is_frequency_job, _numerical_cartesian_gradient, _optimize_semiempirical_numerical, _semiempirical_energy_for_coords, abs, append_log, array, asarray, atom_coords, build_result_dict, calculate_frontier_orbitals, copy, dip_moment. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_pyscf_semiempirical(spec, config, mol_dir, stop_event, progress_log):
    try:
        ensure_python_package("pyscf.semiempirical", "pyscf[semiempirical]")
        import pyscf
        from pyscf import semiempirical
    except Exception as exc:
        raise RuntimeError(
            f"Ekstensi semiempirik PySCF belum tersedia dan instalasi otomatis gagal:
```

```
{exc}"
        ) from exc

    rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
    atoms = mol_to_pyscf_atoms(rdmol)
    method = config["method"]
    if _is_frequency_job(config["job"]):
        raise RuntimeError(
            "Frequency untuk ekstensi PySCF Semiempirical AM1/MINDO/3 tidak dipaksakan
dengan Hessian palsu. "
            "Pilih backend MOPAC CLI untuk vibrasi semiempirik (FORCE), atau PySCF
HF/DFT/xTB/ORCA."
        )
    charge = int(spec["charge"])
    spin = int(spec["multiplicity"] - 1)
    symbols = [a[0] for a in atoms]
    coords0 = np.asarray([a[1] for a in atoms], dtype=float)

    progress_path = Path(progress_log)
    append_log(
        progress_path,
```

A.70 build_result_dict()

> **Lokasi dan signature:** Baris 2053–2131; build_result_dict(spec, config, final_rdmol, e_initial, e_final, mf, frontier, dip_vec, dip_total, grad_rms, grad_max, orbitals_df, charges_df, formats, coords, frequencies, backend_note)

Fungsi `build_result_dict` berperan untuk membangun representasi, geometri, visualisasi, atau file keluaran. Jumlah argumen posisi yang terdeteksi adalah 17; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetAtomPosition, GetAtoms, GetConformer, GetNumAtoms, GetSymbol, append, array, bool, energy_units, enumerate, export_bonded_formats, float, get, getattr, hasattr, int, len, molecule_identifiers. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def build_result_dict(spec, config, final_rdmol, e_initial, e_final, mf=None,
                      frontier=None, dip_vec=None, dip_total=None,
                      grad_rms=None, grad_max=None, orbitals_df=None,
                      charges_df=None, formats=None, coords=None, frequencies=None,
                      backend_note=""):
    ids = molecule_identifiers(final_rdmol)
    units = energy_units(e_final)
    delta_units = energy_units(e_final - e_initial if e_initial is not None and e_final is
not None else None)
    if coords is None:
        conf = final_rdmol.GetConformer()
        coords = np.array([[conf.GetAtomPosition(i).x, conf.GetAtomPosition(i).y,
conf.GetAtomPosition(i).z]
                           for i in range(final_rdmol.GetNumAtoms())])
    coords_rows = []
    for i, atom in enumerate(final_rdmol.GetAtoms()):
        coords_rows.append({
            "Atom": i + 1, "Unsur": atom.GetSymbol(),
            "X (Å)": float(coords[i, 0]), "Y (Å)": float(coords[i, 1]), "Z (Å)":
float(coords[i, 2]),
        })
    if frontier is None:
        frontier = {"homo_index": None, "lumo_index": None, "homo_ev": None, "lumo_ev":
None, "gap_ev": None}
    if dip_vec is None:
        dip_vec = [None, None, None]
    if formats is None:
        formats = export_bonded_formats(final_rdmol, spec["name"])
    if frequencies is None:
        frequencies = []
```

A.71 _is_frequency_job()

Lokasi dan signature: Baris 2134–2135; _is_frequency_job(job_name)

Fungsi `_is_frequency_job` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: str. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _is_frequency_job(job_name):
    return "Frequency" in str(job_name)
```

A.72 _is_optimization_job()

Lokasi dan signature: Baris 2138–2139; _is_optimization_job(job_name)

Fungsi `_is_optimization_job` berperan untuk mengelola siklus hidup job dan status UI. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: startswith, str. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _is_optimization_job(job_name):
    return str(job_name).startswith("Geometry Optimization")
```

A.73 _pyscf_hessian_to_frequencies()

Lokasi dan signature: Baris 2142–2180; _pyscf_hessian_to_frequencies(mol, hessian)

Fungsi `_pyscf_hessian_to_frequencies` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Konversi Hessian PySCF (Eh/Bohr^2) menjadi frekuensi harmonik cm^-1.

Interaksi utama yang terdeteksi: abs, argsort, asarray, atom_coords, atom_mass_list, average, bool, dot, eigvalsh, enumerate, eye, float, int, len, max, min, ones, outer. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _pyscf_hessian_to_frequencies(mol, hessian):
    """Konversi Hessian PySCF (Eh/Bohr^2) menjadi frekuensi harmonik cm^-1."""
    h = np.asarray(hessian, dtype=float)
    n = int(mol.natm)
    if h.shape == (n, n, 3, 3):
        h = h.transpose(0, 2, 1, 3).reshape(3 * n, 3 * n)
    elif h.shape != (3 * n, 3 * n):
        h = h.reshape(3 * n, 3 * n)
    h = 0.5 * (h + h.T)

    masses = np.asarray(mol.atom_mass_list(), dtype=float)
    m3 = np.repeat(masses, 3)
    mw_h = h / np.sqrt(np.outer(m3, m3))
    eigvals = np.linalg.eigvalsh(mw_h)

    if n == 1:
        remove_n = 3
    elif n == 2:
        remove_n = 5
    else:
        coords = np.asarray(mol.atom_coords(unit="Angstrom"), dtype=float)
        com = np.average(coords, axis=0, weights=masses)
        inertia = np.zeros((3, 3), dtype=float)
        for mass, r in zip(masses, coords - com):
            inertia += mass * ((np.dot(r, r) * np.eye(3)) - np.outer(r, r))
        moments = np.linalg.eigvalsh(inertia)
```

A.74 calculate_pyscf_frequencies()

> **Lokasi dan signature:** Baris 2183–2191; calculate_pyscf_frequencies(mf, mol, progress_log, return_hessian)

Fungsi `calculate_pyscf_frequencies` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 4; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Hessian, Path, RuntimeError, _pyscf_hessian_to_frequencies, append_log, asarray, kernel, len, sum. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def calculate_pyscf_frequencies(mf, mol, progress_log, return_hessian=False):
    append_log(Path(progress_log), "Menghitung Hessian/frekuensi harmonik PySCF.")
    try:
        hess = np.asarray(mf.Hessian().kernel(), dtype=float)
    except Exception as exc:
        raise RuntimeError(f"Hessian tidak tersedia untuk metode/referensi PySCF ini:
{exc}") from exc
    rows = _pyscf_hessian_to_frequencies(mol, hess)
    append_log(Path(progress_log), f"Frequency selesai: {len(rows)} mode; imaginary={sum(1
for r in rows if r['Imaginary'])}.")
    return (rows, hess) if return_hessian else rows
```

A.75 run_pyscf_calculation()

Lokasi dan signature: Baris 2194–2346; run_pyscf_calculation(spec, config, mol_dir, stop_event, progress_log)

Fungsi `run_pyscf_calculation` berperan untuk menjalankan atau menyiapkan komputasi PySCF. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 1 pernyataan return dan 5 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: MP2, MolFromMolBlock, Path, RuntimeError, StopRequested, ValueError, _is_frequency_job, abs, append_log, apply_selected_addon_pyscf, array, asarray, atom_coords, bool, build_result_dict, calculate_frontier_orbitals, calculate_pyscf_frequencies, dip_moment. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_pyscf_calculation(spec, config, mol_dir, stop_event, progress_log):
    method = config["method"]
    if method not in PYSCF_METHODS:
        raise ValueError(f"Metode PySCF tidak dikenal: {method}")
    if config["job"].startswith("Geometry Optimization") and not
PYSCF_METHODS[method]["opt"]:
        raise ValueError(f"{method} pada aplikasi ini dibatasi untuk Single Point agar
hasil tidak menyesatkan.")

    rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
    if rdmol is None:
        raise ValueError("MolBlock input gagal dibaca kembali oleh worker.")
    atoms = mol_to_pyscf_atoms(rdmol)
    charge = int(spec["charge"])
    spin = int(spec["multiplicity"] - 1)
    basis = config["basis"]

    progress_path = Path(progress_log)
    chkfile = mol_dir / f"{safe_slug(spec['name'])}.chk"

    with open(progress_path, "a", encoding="utf-8") as logh:
        mol0 = make_pyscf_molecule(
            atoms, basis=basis, charge=charge, spin=spin,
            max_memory_mb=config["max_memory_mb"], log_handle=logh,
        )
        mf0 = make_pyscf_method(
```

```
            mol0, method,
            grid_level=config["grid_level"],
```

A.76 parse_xyz_text()

Lokasi dan signature: Baris 2352–2362; parse_xyz_text(xyz_text)

Fungsi `parse_xyz_text` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: append, asarray, float, int, rstrip, split, splitlines, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def parse_xyz_text(xyz_text):
    lines = [x.rstrip() for x in xyz_text.strip().splitlines()]
    n = int(lines[0].strip())
    atom_lines = lines[2:2 + n]
    symbols = []
    coords = []
    for line in atom_lines:
        p = line.split()
        symbols.append(p[0])
        coords.append([float(p[1]), float(p[2]), float(p[3])])
    return symbols, np.asarray(coords, dtype=float)
```

A.77 run_process_interruptible()

Lokasi dan signature: Baris 2365–2384; run_process_interruptible(cmd, cwd, stdout_path, stop_event, env)

Fungsi `run_process_interruptible` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 0 pernyataan return dan 2 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Popen, RuntimeError, StopRequested, is_set, kill, open, poll, sleep, str, terminate, wait. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_process_interruptible(cmd, cwd, stdout_path, stop_event, env=None):
    with open(stdout_path, "a", encoding="utf-8") as out:
        proc = subprocess.Popen(
            cmd, cwd=str(cwd), stdout=out, stderr=subprocess.STDOUT,
            text=True, env=env,
        )
        while proc.poll() is None:
            if stop_event.is_set():
                try:
                    proc.terminate()
```

```
                    proc.wait(timeout=5)
                except Exception:
                    try:
                        proc.kill()
                    except Exception:
                        pass
                raise StopRequested("Proses eksternal dihentikan oleh pengguna.")
            time.sleep(0.5)
        if proc.returncode != 0:
            raise RuntimeError(f"Program eksternal gagal dengan exit code
{proc.returncode}. Lihat progress.log.")
```

A.78 run_xtb_calculation()

> **Lokasi dan signature:** Baris 2387–2450; run_xtb_calculation(spec, config, mol_dir, stop_event, progress_log)

Fungsi `run_xtb_calculation` berperan untuk menjalankan workflow xTB. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 1 pernyataan return dan 3 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetNumAtoms, MolFromMolBlock, Path, RuntimeError, _is_frequency_job, abs, append, append_log, bool, build_result_dict, enumerate, exists, export_bonded_formats, findall, float, join, len, mol_to_xyz. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_xtb_calculation(spec, config, mol_dir, stop_event, progress_log):
    xtb = shutil.which("xtb")
    if not xtb:
        raise RuntimeError("Executable 'xtb' tidak ditemukan pada PATH.")

    rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
    inp_xyz = mol_to_xyz(rdmol, spec["name"])
    xyz_path = mol_dir / "input.xyz"
    xyz_path.write_text(inp_xyz, encoding="utf-8")

    gfn = {"GFN2-xTB": "2", "GFN1-xTB": "1", "GFN0-xTB": "0"}[config["method"]]
    cmd = [xtb, str(xyz_path), "--gfn", gfn, "--chrg", str(spec["charge"]), "--uhf",
str(spec["multiplicity"] - 1)]
    if config["job"] == "Geometry Optimization + Frequency":
        cmd += ["--ohess"]
    elif config["job"] == "Frequency Analysis":
        cmd += ["--hess"]
    elif config["job"].startswith("Geometry Optimization"):
        cmd += ["--opt"]
    append_log(progress_log, "Menjalankan: " + " ".join(cmd))
    run_process_interruptible(cmd, mol_dir, progress_log, stop_event)

    log_text = Path(progress_log).read_text(encoding="utf-8", errors="replace")
    matches = re.findall(r"TOTAL ENERGY\s+(-?\d+\.\d+(?:[Ee][+-]?\d+)?)", log_text)
    if not matches:
        matches = re.findall(r"total energy\s+(-?\d+\.\d+(?:[Ee][+-]?\d+)?)", log_text,
flags=re.I)
    if not matches:
```

A.79 run_orca_calculation()

> **Lokasi dan signature:** Baris 2453–2535; run_orca_calculation(spec, config, mol_dir, stop_event, progress_log)

Fungsi `run_orca_calculation` berperan untuk menjalankan workflow ORCA. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 1 pernyataan return dan 2 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetAtomPosition, GetAtoms, GetConformer, GetNumAtoms, GetSymbol, MolFromMolBlock, Path, RuntimeError, _is_frequency_job, append, append_log, array, bool, build_result_dict, endswith, enumerate, exists, export_bonded_formats. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_orca_calculation(spec, config, mol_dir, stop_event, progress_log):
    orca = shutil.which("orca")
    if not orca:
        raise RuntimeError("Executable 'orca' tidak ditemukan pada PATH.")

    rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
    conf = rdmol.GetConformer()
    method = config["method"]
    basis = config["basis"]
    if config["job"] == "Geometry Optimization + Frequency":
        job_keyword = "Opt Freq"
    elif config["job"] == "Frequency Analysis":
        job_keyword = "Freq"
    elif config["job"].startswith("Geometry Optimization"):
        job_keyword = "Opt"
    elif config["job"] == "Single Point + Gradient":
        job_keyword = "EnGrad"
    else:
        job_keyword = "SP"
    mult = int(spec["multiplicity"])
    charge = int(spec["charge"])
    nprocs = int(config["threads"])
    maxcore = max(100, int(config["max_memory_mb"] // max(1, nprocs)))

    # Nama metode yang aman untuk input ORCA.
    method_orca = {"CAM-B3LYP": "CAM-B3LYP", "HF": "HF"}.get(method, method)
```

A.80 _mopac_spin_keywords()

Lokasi dan signature: Baris 2542–2553; _mopac_spin_keywords(multiplicity)

Fungsi `_mopac_spin_keywords` berperan untuk menulis, menjalankan, atau memparsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: append, int. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_spin_keywords(multiplicity):
    labels = {
        1: "SINGLET", 2: "DOUBLET", 3: "TRIPLET", 4: "QUARTET",
        5: "QUINTET", 6: "SEXTET", 7: "SEPTET",
    }
    mult = int(multiplicity)
    words = []
    if mult in labels:
        words.append(labels[mult])
    if mult > 1:
        words.append("UHF")
    return words
```

A.81 _write_mopac_input()

Lokasi dan signature: Baris 2556–2578; _write_mopac_input(path, rdmol, method, charge, multiplicity)

Fungsi `_write_mopac_input` berperan untuk menulis, menjalankan, atau memparsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetAtomPosition, GetAtoms, GetConformer, GetSymbol, Path, _mopac_spin_keywords, append, enumerate, extend, int, join, str, write_text. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _write_mopac_input(path, rdmol, method, charge, multiplicity, *, optimize=False,
                       force=False, gradient=False, maxsteps=100, title="QM job"):
    keywords = [method, "AUX", "PRECISE", f"CHARGE={int(charge)}"]
    keywords.extend(_mopac_spin_keywords(multiplicity))
    if optimize:
        keywords += [f"CYCLES={int(maxsteps)}"]
    else:
        keywords += ["1SCF"]
    if force:
        keywords += ["FORCE", "LET"]
    if gradient:
        keywords += ["GRADIENTS"]

    conf = rdmol.GetConformer()
    flag = 1 if optimize else 0
    lines = [" ".join(keywords), str(title), "Generated by app_qm_streamlit_v2_2"]
    for i, atom in enumerate(rdmol.GetAtoms()):
        pt = conf.GetAtomPosition(i)
        lines.append(
```

```
            f"{atom.GetSymbol():2s} {pt.x: .10f} {flag:d} {pt.y: .10f} {flag:d} {pt.z:
.10f} {flag:d}"
        )
    lines.append("")
    Path(path).write_text("\n".join(lines), encoding="utf-8")
```

A.82 _mopac_number()

Lokasi dan signature: Baris 2581–2583; _mopac_number(text)

Fungsi `_mopac_number` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Konversi angka MOPAC, termasuk notasi Fortran D+/-xx.

Interaksi utama yang terdeteksi: float, replace, str, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_number(text):
    """Konversi angka MOPAC, termasuk notasi Fortran D+/-xx."""
    return float(str(text).strip().replace("D", "E").replace("d", "e"))
```

A.83 _mopac_aux_scalar()

Lokasi dan signature: Baris 2589–2604; _mopac_aux_scalar(text, key)

Fungsi `_mopac_aux_scalar` berperan untuk menulis, menjalankan, atau memparsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Ambil scalar AUX secara toleran terhadap spasi, case, dan format MOPAC.

Interaksi utama yang terdeteksi: _mopac_number, escape, finditer, group, list. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_aux_scalar(text, key):
    """Ambil scalar AUX secara toleran terhadap spasi, case, dan format MOPAC."""
    if not text:
        return None
    patterns = [
        rf"^\s*{re.escape(key)}\s*=\s*({_MOPAC_NUMBER_RE})",
        rf"^\s*{re.escape(key)}\s*:\s*({_MOPAC_NUMBER_RE})",
    ]
    for pattern in patterns:
        matches = list(re.finditer(pattern, text, flags=re.M | re.I))
        if matches:
            try:
                return _mopac_number(matches[-1].group(1))
            except Exception:
                pass
    return None
```

A.84 _mopac_aux_scalar_any()

Lokasi dan signature: Baris 2607–2612; _mopac_aux_scalar_any(text, keys)

Fungsi `_mopac_aux_scalar_any` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: _mopac_aux_scalar. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_aux_scalar_any(text, keys):
    for key in keys:
        value = _mopac_aux_scalar(text, key)
        if value is not None:
            return value, key
    return None, None
```

A.85 _mopac_coords_from_aux()

Lokasi dan signature: Baris 2615–2631; _mopac_coords_from_aux(text, natm)

Fungsi `_mopac_coords_from_aux` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: _mopac_number, append, asarray, end, escape, findall, finditer, len, list, reshape, reversed, search, start. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_coords_from_aux(text, natm):
    keys = ["ATOM_X_OPT:ANGSTROMS", "ATOM_X_UPDATED:ANGSTROMS", "ATOM_X:ANGSTROMS"]
    for key in keys:
        matches = list(re.finditer(rf"^\s*{re.escape(key)}\[\d+\]\s*=\s*$", text,
flags=re.M | re.I))
        for m in reversed(matches):
            tail = text[m.end():]
            stop = re.search(r"^\s*[A-Z][A-Z0-9_().:/+-]*(?:\[\d+\])?\s*=", tail,
flags=re.M | re.I)
            chunk = tail[:stop.start()] if stop else tail
            vals = []
            for tok in re.findall(_MOPAC_NUMBER_RE, chunk):
                try:
                    vals.append(_mopac_number(tok))
                except Exception:
                    pass
            if len(vals) >= 3 * natm:
                return np.asarray(vals[:3 * natm], dtype=float).reshape(natm, 3)
    return None
```

A.86 _mopac_coords_from_out()

Lokasi dan signature: Baris 2634–2653; _mopac_coords_from_out(text, natm)

Fungsi `_mopac_coords_from_out` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: _mopac_number, append, asarray, finditer, group, len, match, max, reversed, splitlines, start. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_coords_from_out(text, natm):
    # Fallback: ambil tabel CARTESIAN COORDINATES terakhir yang lengkap.
    starts = [m.start() for m in re.finditer(r"CARTESIAN COORDINATES", text, flags=re.I)]
    for start in reversed(starts):
        chunk = text[start:start + max(4000, natm * 160)]
        rows = []
        for line in chunk.splitlines():
            m = re.match(
                rf"^\s*\d+\s+([A-Za-
z]{{1,3}})\s+({_MOPAC_NUMBER_RE})\s+({_MOPAC_NUMBER_RE})\s+({_MOPAC_NUMBER_RE})(?:\s+.*)?$
",
                line,
```

```
                flags=re.I,
            )
            if m:
                try:
                    rows.append([_mopac_number(m.group(2)), _mopac_number(m.group(3)),
_mopac_number(m.group(4))])
                except Exception:
                    continue
                if len(rows) == natm:
                    return np.asarray(rows, dtype=float)
    return None
```

A.87 _mopac_total_energy_details()

Lokasi dan signature: Baris 2656–2730;
_mopac_total_energy_details(out_text, aux_text)

Fungsi `_mopac_total_energy_details` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 7 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Kembalikan (energy_hartree, source, details) tanpa menyamakan HOF dengan total energy. MOPAC normalnya menulis TOTAL ENERGY dalam eV pada .out. File AUX yang lebih machine-readable lazimnya menyediakan ENERGY_ELECTRONIC:EV dan ENERGY_NUCLEAR:EV. Beberapa versi/build menggunakan nama TOTAL_ENERGY:EV.

Interaksi utama yang terdeteksi: _last_out_scalar, _mopac_aux_scalar_any, _mopac_number, finditer, group, list. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_total_energy_details(out_text, aux_text=""):
    """Kembalikan (energy_hartree, source, details) tanpa menyamakan HOF dengan total
energy.

    MOPAC normalnya menulis TOTAL ENERGY dalam eV pada .out. File AUX yang lebih
    machine-readable lazimnya menyediakan ENERGY_ELECTRONIC:EV dan
    ENERGY_NUCLEAR:EV. Beberapa versi/build menggunakan nama TOTAL_ENERGY:EV.
    """
    details = {}

    # 1) TOTAL ENERGY dari output utama: toleran terhadap '=' / ':' / whitespace.
    out_patterns = [
        rf"TOTAL\s+ENERGY\s*(?:=|:)\s*({_MOPAC_NUMBER_RE})\s*(?:EV|ELECTRON\s*VOLTS?)",
        rf"TOTAL\s+ENERGY\s+({_MOPAC_NUMBER_RE})\s*(?:EV|ELECTRON\s*VOLTS?)",
        rf"TOTAL_ENERGY(?::EV)?\s*=\s*({_MOPAC_NUMBER_RE})",
    ]
    for pattern in out_patterns:
        matches = list(re.finditer(pattern, out_text or "", flags=re.I | re.M))
        if matches:
            value_ev = _mopac_number(matches[-1].group(1))
            details["total_energy_ev"] = value_ev
            return value_ev / HARTREE_TO_EV, "OUT:TOTAL ENERGY", details

    # 2) TOTAL ENERGY langsung dari AUX, bila disediakan build MOPAC tersebut.
    aux_total, aux_total_key = _mopac_aux_scalar_any(
        aux_text,
        ["TOTAL_ENERGY:EV", "ENERGY_TOTAL:EV", "TOTAL ENERGY:EV", "TOTAL_ENERGY"],
```

A.88 _last_out_scalar()

Lokasi dan signature: Baris 2706–2714; _last_out_scalar(labels)

Fungsi `_last_out_scalar` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: _mopac_number, finditer, group, list. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _last_out_scalar(labels):
        for label in labels:
            matches = list(re.finditer(
                rf"{label}\s*(?:=|:)\s*({_MOPAC_NUMBER_RE})\s*(?:EV|ELECTRON\s*VOLTS?)",
                out_text or "", flags=re.I | re.M,
            ))
            if matches:
                return _mopac_number(matches[-1].group(1)), label
        return None, None
```

A.89 _mopac_total_energy_h()

Lokasi dan signature: Baris 2733–2736; _mopac_total_energy_h(out_text, aux_text)

Fungsi `_mopac_total_energy_h` berperan untuk menulis, menjalankan, atau memparsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Kompatibilitas fungsi lama: hanya kembalikan energi dalam Hartree atau None.

Interaksi utama yang terdeteksi: _mopac_total_energy_details. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.

- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_total_energy_h(out_text, aux_text=""):
    """Kompatibilitas fungsi lama: hanya kembalikan energi dalam Hartree atau None."""
    energy_h, _source, _details = _mopac_total_energy_details(out_text, aux_text)
    return energy_h
```

A.90 _mopac_heat_of_formation_kcal()

Lokasi dan signature: Baris 2739–2755;
_mopac_heat_of_formation_kcal(out_text, aux_text)

Fungsi `_mopac_heat_of_formation_kcal` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Ambil heat of formation sebagai properti terpisah; jangan diklaim sebagai Hartree.

Interaksi utama yang terdeteksi: _mopac_aux_scalar_any, _mopac_number, finditer, group, list. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_heat_of_formation_kcal(out_text, aux_text=""):
    """Ambil heat of formation sebagai properti terpisah; jangan diklaim sebagai
Hartree."""
    hof, _key = _mopac_aux_scalar_any(
```

```
        aux_text,
        ["HEAT_OF_FORMATION:KCAL/MOL", "HEAT_OF_FORM_UPDATED:KCAL/MOL"],
    )
    if hof is not None:
        return hof
    patterns = [
        rf"HEAT\s+OF\s+FORMATION\s*(?:=|:)\s*({_MOPAC_NUMBER_RE})\s*KCAL/MOL",
        rf"FINAL\s+HEAT\s+OF\s+FORMATION\s*(?:=|:)\s*({_MOPAC_NUMBER_RE})\s*KCAL/MOL",
    ]
    for pattern in patterns:
        matches = list(re.finditer(pattern, out_text or "", flags=re.I | re.M))
        if matches:
            return _mopac_number(matches[-1].group(1))
    return None
```

A.91 _mopac_diagnostic_excerpt()

Lokasi dan signature: Baris 2758–2774;
_mopac_diagnostic_excerpt(out_text, aux_text, max_lines)

Fungsi `_mopac_diagnostic_excerpt` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Ambil pesan MOPAC yang relevan agar error parser tidak menutupi error asli.

Interaksi utama yang terdeteksi: append, compile, join, search, splitlines, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_diagnostic_excerpt(out_text, aux_text="", max_lines=28):
    """Ambil pesan MOPAC yang relevan agar error parser tidak menutupi error asli."""
```

```
    interesting = []
    keywords = re.compile(
        r"ERROR|FATAL|UNRECOGNI[ZS]ED|NOT\s+RECOGNI[ZS]ED|FAILED|FAILURE|"
        r"SCF\s+FIELD\s+WAS\s+NOT|NO\s+PARAMETER|PARAMETER.*MISSING|"
        r"TOTAL\s+ENERGY|HEAT\s+OF\s+FORMATION|ELECTRONIC\s+ENERGY|CORE-CORE",
        flags=re.I,
    )
    for label, text in (("OUT", out_text), ("AUX", aux_text)):
        for line in (text or "").splitlines():
            if keywords.search(line):
                interesting.append(f"[{label}] {line.strip()}")
    if not interesting:
        tail = [ln.strip() for ln in (out_text or "").splitlines() if ln.strip()][-
max_lines:]
        interesting = [f"[OUT] {ln}" for ln in tail]
    return "\n".join(interesting[-max_lines:])
```

A.92 _mopac_frequencies()

Lokasi dan signature: Baris 2777–2805; _mopac_frequencies(out_text, aux_text)

Fungsi `_mopac_frequencies` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: _mopac_number, abs, append, bool, end, enumerate, findall, finditer, float, group, search, start. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.

- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _mopac_frequencies(out_text, aux_text=""):
    vals = []
    for m in re.finditer(r"\bFREQ\.\s*(-?\d+(?:\.\d+)?)", out_text, flags=re.I):
        try:
            vals.append(float(m.group(1)))
        except Exception:
            pass
    if not vals:
        # Beberapa versi AUX menulis vector yang mengandung FREQ pada nama field.
        for m in re.finditer(r"^[^\n=]*FREQ[^\n=]*\[\d+\]\s*=\s*$", aux_text, flags=re.I |
re.M):
            tail = aux_text[m.end():]
            stop = re.search(r"^\s*[A-Z][A-Z0-9_().:/+-]*(?:\[\d+\])?\s*=", tail,
flags=re.M)
            chunk = tail[:stop.start()] if stop else tail
            for tok in re.findall(r"[-+]?\d*\.?\d+(?:[EeDd][-+]?\d+)?", chunk):
                try:
                    vals.append(_mopac_number(tok))
                except Exception:
                    pass
            if vals:
                break
    # De-duplicate sequential repeats sometimes present in verbose descriptions.
    cleaned = []
    for v in vals:
        if not cleaned or abs(v - cleaned[-1]) > 1.0e-8:
            cleaned.append(v)
    return [
```

A.93 _run_one_mopac()

> **Lokasi dan signature:** Baris 2808–2844; _run_one_mopac(mopac, input_path, mol_dir, stop_event, progress_log)

Fungsi `_run_one_mopac` berperan untuk menulis, menjalankan, atau mem-parsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, RuntimeError, append_log, exists, glob, lower, next, read_text, run_process_interruptible, sorted, stat, str, with_suffix. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _run_one_mopac(mopac, input_path, mol_dir, stop_event, progress_log):
    append_log(Path(progress_log), "Menjalankan MOPAC: " + str(input_path.name))
    run_process_interruptible([mopac, str(input_path)], mol_dir, progress_log, stop_event)

    # MOPAC normalnya menghapus ekstensi .mop/.dat saat membentuk nama output.
    stem = input_path.with_suffix("")
    candidates_out = [
        Path(str(stem) + ext) for ext in (".out", ".OUT", ".Out")
    ]
    candidates_aux = [
        Path(str(stem) + ext) for ext in (".aux", ".AUX", ".Aux")
    ]
    out_path = next((p for p in candidates_out if p.exists()), None)
    aux_path = next((p for p in candidates_aux if p.exists()), None)

    # Fallback untuk build/wrapper yang memberi variasi kapitalisasi nama file.
    if out_path is None:
        matches = sorted(mol_dir.glob(stem.name + ".*"), key=lambda p: p.stat().st_mtime
if p.exists() else 0, reverse=True)
        out_path = next((p for p in matches if p.suffix.lower() == ".out"), None)
    if aux_path is None:
        matches = sorted(mol_dir.glob(stem.name + ".*"), key=lambda p: p.stat().st_mtime
if p.exists() else 0, reverse=True)
        aux_path = next((p for p in matches if p.suffix.lower() == ".aux"), None)

    out_text = out_path.read_text(encoding="utf-8", errors="replace") if out_path and
out_path.exists() else ""
    aux_text = aux_path.read_text(encoding="utf-8", errors="replace") if aux_path and
aux_path.exists() else ""
    if not out_text and not aux_text:
```

A.94 run_mopac_calculation()

Lokasi dan signature: Baris 2847–2943; run_mopac_calculation(spec, config, mol_dir, stop_event, progress_log)

Fungsi `run_mopac_calculation` berperan untuk menulis, menjalankan, atau memparsing workflow MOPAC. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 1 pernyataan return dan 4 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: GetAtomPosition, GetConformer, GetNumAtoms, MolFromMolBlock, Path, RuntimeError, ValueError, _is_frequency_job, _is_optimization_job, _mopac_aux_scalar, _mopac_coords_from_aux, _mopac_coords_from_out, _mopac_diagnostic_excerpt, _mopac_frequencies, _mopac_heat_of_formation_kcal, _mopac_total_energy_details, _run_one_mopac, _write_mopac_input. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def run_mopac_calculation(spec, config, mol_dir, stop_event, progress_log):
    mopac = find_mopac_executable()
    if not mopac:
        raise RuntimeError("Executable MOPAC tidak ditemukan pada PATH.")

    method = config["method"]
    if method not in MOPAC_METHODS:
        raise ValueError(f"Metode MOPAC tidak dikenal: {method}")
```

```
rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
if rdmol is None:
    raise ValueError("MolBlock input gagal dibaca untuk MOPAC.")
natm = rdmol.GetNumAtoms()
job = config["job"]
optimize = _is_optimization_job(job)
wants_freq = _is_frequency_job(job)

primary = mol_dir / "mopac_job.mop"
_write_mopac_input(
    primary, rdmol, method, spec["charge"], spec["multiplicity"],
    optimize=optimize,
    force=(wants_freq and not optimize),
    gradient=(job == "Single Point + Gradient"),
    maxsteps=config["max_opt_steps"],
    title=f"{spec['name']} | {method} | {job}",
)
```

A.95 save_result_files()

Lokasi dan signature: Baris 2949–2985; save_result_files(result, mol_dir)

Fungsi `save_result_files` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: DataFrame, get, items, safe_slug, to_csv, write_json_atomic, write_text. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.

- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def save_result_files(result, mol_dir: Path):
    base = safe_slug(result["name"])
    project_base = safe_slug(result.get("project_name", "QM_Project"))
    project_prefix = f"{project_base}__{base}"

    # Nama file lama tetap dipertahankan agar workflow yang sudah ada tidak rusak.
    (mol_dir / f"{base}_final.xyz").write_text(result.get("xyz", ""), encoding="utf-8")
    (mol_dir / f"{base}_final.mol").write_text(result.get("mol", ""), encoding="utf-8")
    (mol_dir / f"{base}_final.sdf").write_text(result.get("sdf", ""), encoding="utf-8")
    (mol_dir / f"{base}_final.pdb").write_text(result.get("pdb", ""), encoding="utf-8")
    pd.DataFrame(result.get("coords", [])).to_csv(mol_dir / f"{base}_coordinates.csv",
index=False)
    pd.DataFrame(result.get("orbitals", [])).to_csv(mol_dir / f"{base}_orbitals.csv",
index=False)
    pd.DataFrame(result.get("charges", [])).to_csv(mol_dir / f"{base}_mulliken.csv",
index=False)
    pd.DataFrame(result.get("frequencies", [])).to_csv(mol_dir /
f"{base}_frequencies.csv", index=False)
    pd.DataFrame(result.get("bond_lengths", [])).to_csv(mol_dir /
f"{base}_all_bond_lengths.csv", index=False)
    pd.DataFrame(result.get("bond_angles", [])).to_csv(mol_dir /
f"{base}_all_bond_angles.csv", index=False)
    pd.DataFrame(result.get("dihedral_angles", [])).to_csv(mol_dir /
f"{base}_all_dihedral_angles.csv", index=False)
    pd.DataFrame(result.get("optimization_geometry_energies", [])).to_csv(mol_dir /
f"{base}_optimization_geometry_energies.csv", index=False)
    pd.DataFrame(result.get("optimization_energy_history", [])).to_csv(mol_dir /
f"{base}_optimization_energy_history.csv", index=False)
    pd.DataFrame(result.get("qsar_qspr_descriptors", [])).to_csv(mol_dir /
f"{base}_qsar_qspr_descriptors.csv", index=False)
    pd.DataFrame(result.get("addon_results", [])).to_csv(mol_dir /
f"{base}_addon_results.csv", index=False)
    pd.DataFrame(result.get("thermochemistry", [])).to_csv(mol_dir /
f"{base}_thermochemistry.csv", index=False)

    compact = {k: v for k, v in result.items() if k not in {"xyz", "mol", "sdf", "pdb",
"coords", "orbitals", "charges"}}
    write_json_atomic(mol_dir / f"{base}_result.json", compact)
```

A.96 write_summary_log()

Lokasi dan signature: Baris 2988–3028; write_summary_log(run_dir, config, results, errors, stopped)

Fungsi `write_summary_log` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: append, get, join, now_text, write_text. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def write_summary_log(run_dir: Path, config, results, errors, stopped=False):
    lines = [
        "QM COMPUTATION SUMMARY",
        "=" * 72,
        f"Time: {now_text()}",
        f"Project: {config.get('project_name', 'QM_Project')}",
        f"Program QM: {config['program_qm']}",
        f"Method: {config['method']}",
        f"Basis: {config.get('basis', 'Tidak digunakan')}",
        f"Job: {config['job']}",
        f"Optimizer: {config.get('optimizer', '-')}",
        f"Add-On: {config.get('addon', 'None')}",
        f"SCF tolerance: {config['scf_conv_tol']:.0e}",
        f"CPU threads: {config.get('threads', 1)}",
        f"GPU enabled: {config.get('use_gpu', False)}",
        f"GPU Threads: {config.get('gpu_threads', 0)}",
        f"Stopped by user: {stopped}",
        "",
    ]
    for r in results:
        e = r["energy"]
        lines += [
            f"[{r['name']}]",
            f"Formula: {r['formula']}",
            f"SMILES: {r['smiles']}",
            f"InChI: {r['inchi']}",
```

A.97 job_worker()

Lokasi dan signature: Baris 3031–3114; job_worker(run_dir_str, specs, config, stop_event)

Fungsi `job_worker` berperan untuk mengelola siklus hidup job dan status UI. Jumlah argumen posisi yang terdeteksi adalah 4; tubuh fungsi memiliki 0 pernyataan return dan 3 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, StopRequested, ValueError, append, append_log, enrich_result_common, enumerate, format_exc, get, int, is_set, len, mkdir, now_text, num_threads, run_mopac_calculation, run_orca_calculation, run_pyscf_calculation. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def job_worker(run_dir_str, specs, config, stop_event):
    run_dir = Path(run_dir_str)
    progress_log = run_dir / "progress.log"
    error_log = run_dir / "error.log"
    status_file = run_dir / "status.json"
    results_file = run_dir / "results.json"
    results = []
    errors = []

    def status(state, message, current=0, total=None):
        write_json_atomic(status_file, {
```

```
            "state": state,
            "message": message,
            "current": current,
            "total": len(specs) if total is None else total,
            "time": now_text(),
        })

    try:
        append_log(progress_log, f"Job dimulai. Project={config.get('project_name',
'QM_Project')}")
        append_log(progress_log, f"CPU threads={config.get('threads', 1)};
GPU={config.get('use_gpu', False)}; GPU Threads={config.get('gpu_threads', 0)}")
        status("running", "Menyiapkan perhitungan...", 0)
        lib.num_threads(int(config["threads"]))

        for i, spec in enumerate(specs, start=1):
            if stop_event.is_set():
```

A.98 status()

Lokasi dan signature: Baris 3040–3047; status(state, message, current, total)

Fungsi `status` berperan untuk mengelola siklus hidup job dan status UI. Jumlah argumen posisi yang terdeteksi adalah 4; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: len, now_text, write_json_atomic. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def status(state, message, current=0, total=None):
        write_json_atomic(status_file, {
            "state": state,
            "message": message,
            "current": current,
            "total": len(specs) if total is None else total,
            "time": now_text(),
        })
```

A.99 make_spec()

> **Lokasi dan signature:** Baris 3120–3137; make_spec(name, mol, source, random_seed, preopt, charge, multiplicity)

Fungsi `make_spec` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 7; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: MolToMolBlock, formal_charge, int, molecule_identifiers, prepare_molecule_3d. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def make_spec(name, mol, source, random_seed, preopt=True, charge=None, multiplicity=1):
    mol3d = prepare_molecule_3d(mol, random_seed=random_seed, preopt=preopt)
    ids = molecule_identifiers(mol3d)
    if charge is None:
        charge = formal_charge(mol3d)
    molblock = Chem.MolToMolBlock(mol3d)
    return {
        "name": name,
        "source": source,
        "molblock": molblock,
        "formula": ids["formula"],
        "smiles": ids["smiles"],
        "inchi": ids["inchi"],
        "inchikey": ids["inchikey"],
        "mw": ids["mw"],
        "charge": int(charge),
        "multiplicity": int(multiplicity),
    }
```

A.100 _split_named_smiles()

Lokasi dan signature: Baris 3140–3151; _split_named_smiles(line, idx)

Fungsi `_split_named_smiles` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Pisahkan Nama=SMILES tanpa merusak tanda '=' yang merupakan bagian SMILES.

Interaksi utama yang terdeteksi: MolFromSmiles, split, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _split_named_smiles(line: str, idx: int):
    """Pisahkan Nama=SMILES tanpa merusak tanda '=' yang merupakan bagian SMILES."""
    # Bare SMILES yang valid (mis. CC(=O)C) harus diprioritaskan; tanda '=' di
    # dalam ikatan rangkap bukan separator nama.
    if Chem.MolFromSmiles(line) is not None:
        return f"SMILES-{idx}", line
    if "=" in line:
        name, ident = line.split("=", 1)
        name, ident = name.strip(), ident.strip()
        if name and ident:
            return name, ident
    return f"SMILES-{idx}", line
```

A.101 _split_named_inchi()

Lokasi dan signature: Baris 3154–3165; _split_named_inchi(line, idx)

Fungsi `_split_named_inchi` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Pisahkan Nama=InChI=... dengan aman karena InChI sendiri mengandung '='.

Interaksi utama yang terdeteksi: find, len, startswith, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _split_named_inchi(line: str, idx: int):
    """Pisahkan Nama=InChI=... dengan aman karena InChI sendiri mengandung '='."""
```

```
    if line.startswith("InChI="):
        return f"InChI-{idx}", line
    marker = "=InChI="
    pos = line.find(marker)
    if pos > 0:
        name = line[:pos].strip()
        ident = "InChI=" + line[pos + len(marker):].strip()
        if name:
            return name, ident
    return f"InChI-{idx}", line
```

A.102 parse_custom_lines()

> **Lokasi dan signature:** Baris 3168–3185; parse_custom_lines(text, random_seed, preopt)

Fungsi `parse_custom_lines` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Parser kompatibilitas internal dengan separator Nama=identifier.

Interaksi utama yang terdeteksi: _split_named_inchi, _split_named_smiles, append, enumerate, make_spec, molecule_from_identifier, splitlines, startswith, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def parse_custom_lines(text, random_seed, preopt):
    """Parser kompatibilitas internal dengan separator Nama=identifier."""
    specs = []
    errors = []
    for idx, raw in enumerate(text.splitlines(), start=1):
```

```
        line = raw.strip()
        if not line or line.startswith("#"):
            continue
        if line.startswith("InChI=") or "=InChI=" in line:
            name, ident = _split_named_inchi(line, idx)
        else:
            name, ident = _split_named_smiles(line, idx)
        try:
            mol = molecule_from_identifier(ident)
            specs.append(make_spec(name, mol, "SMILES/InChI", random_seed, preopt=preopt))
        except Exception as exc:
            errors.append(f"Baris {idx}: {exc}")
    return specs, errors
```

A.103 parse_smiles_lines()

Lokasi dan signature: Baris 3188–3206; parse_smiles_lines(text, random_seed, preopt)

Fungsi `parse_smiles_lines` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Parser khusus SMILES dengan format Nama=SMILES.

Interaksi utama yang terdeteksi: MolFromSmiles, ValueError, _split_named_smiles, append, enumerate, make_spec, splitlines, startswith, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def parse_smiles_lines(text, random_seed, preopt):
    """Parser khusus SMILES dengan format Nama=SMILES."""
    specs, errors = [], []
```

```
    for idx, raw in enumerate(text.splitlines(), start=1):
        line = raw.strip()
        if not line or line.startswith("#"):
            continue
        name, ident = _split_named_smiles(line, idx)
        if ident.startswith("InChI="):
            errors.append(f"Baris SMILES {idx}: terdeteksi InChI. Masukkan pada kotak
InChI, bukan kotak SMILES.")
            continue
        try:
            mol = Chem.MolFromSmiles(ident)
            if mol is None:
                raise ValueError(f"SMILES tidak valid: {ident}")
            specs.append(make_spec(name, mol, "SMILES", random_seed, preopt=preopt))
        except Exception as exc:
            errors.append(f"Baris SMILES {idx}: {exc}")
    return specs, errors
```

A.104 parse_inchi_lines()

Lokasi dan signature: Baris 3209–3230; parse_inchi_lines(text, random_seed, preopt)

Fungsi `parse_inchi_lines` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 1 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Parser khusus InChI dengan format Nama=InChI=... .

Interaksi utama yang terdeteksi: MolFromInchi, ValueError, _split_named_inchi, append, enumerate, make_spec, splitlines, startswith, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def parse_inchi_lines(text, random_seed, preopt):
    """Parser khusus InChI dengan format Nama=InChI=... ."""
    specs, errors = [], []
    for idx, raw in enumerate(text.splitlines(), start=1):
        line = raw.strip()
        if not line or line.startswith("#"):
            continue
        name, ident = _split_named_inchi(line, idx)
        if not ident.startswith("InChI="):
            errors.append(
                f"Baris InChI {idx}: gunakan format Nama=InChI=... atau InChI=...; "
                "SMILES masukkan pada kotak SMILES."
            )
            continue
        try:
            mol = Chem.MolFromInchi(ident)
            if mol is None:
                raise ValueError("InChI tidak valid")
            specs.append(make_spec(name, mol, "InChI", random_seed, preopt=preopt))
        except Exception as exc:
            errors.append(f"Baris InChI {idx}: {exc}")
    return specs, errors
```

A.105 parse_local_paths()

Lokasi dan signature: Baris 3233–3246; parse_local_paths(text, random_seed, preopt)

Fungsi `parse_local_paths` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, append, make_spec, molecule_from_local_path, splitlines, str, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.

- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def parse_local_paths(text, random_seed, preopt):
    specs = []
    errors = []
    for raw in text.splitlines():
        path = raw.strip().strip('"')
        if not path:
            continue
        try:
            mol = molecule_from_local_path(path)
            name = Path(path).stem
            specs.append(make_spec(name, mol, f"File lokal: {path}", random_seed,
preopt=preopt))
        except Exception as exc:
            errors.append(str(exc))
    return specs, errors
```

A.106 load_uploaded_molecules()

Lokasi dan signature: Baris 3249–3256; load_uploaded_molecules()

Fungsi `load_uploaded_molecules` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Baca katalog upload persisten. Urutan file JSON = urutan tampilan terbaru dahulu.

Interaksi utama yang terdeteksi: get, isinstance, read_json_safe. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def load_uploaded_molecules():
    """Baca katalog upload persisten. Urutan file JSON = urutan tampilan terbaru
dahulu."""
    data = read_json_safe(UPLOAD_MOLECULES_FILE, default=[])
    if isinstance(data, dict):
        data = data.get("molecules", [])
    if not isinstance(data, list):
        return []
    return [x for x in data if isinstance(x, dict) and x.get("name")]
```

A.107 _persisted_source_label()

Lokasi dan signature: Baris 3259–3268;
_persisted_source_label(source_type)

Fungsi `_persisted_source_label` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Label singkat sumber katalog persisten; kompatibel dengan upload v2.2.2.

Interaksi utama yang terdeteksi: get, lower, str. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.

- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _persisted_source_label(source_type: str) -> str:
    """Label singkat sumber katalog persisten; kompatibel dengan upload v2.2.2."""
    labels = {
        "smiles": "SMILES",
        "inchi": "InChI",
        "local_file": "File lokal",
        "upload": "Upload",
        "custom": "Input",
    }
    return labels.get(str(source_type or "").lower(), "Input")
```

A.108 _unique_persisted_name()

Lokasi dan signature: Baris 3271–3291; _unique_persisted_name(base_name, entries, source_type, ignore_index)

Fungsi `_unique_persisted_name` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 4; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Buat nama katalog unik tanpa menghilangkan 50 preset bawaan asli.

Interaksi utama yang terdeteksi: _persisted_source_label, add, enumerate, get, keys, set, str, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _unique_persisted_name(base_name: str, entries, source_type: str = "custom",
```

```
ignore_index=None):
    """Buat nama katalog unik tanpa menghilangkan 50 preset bawaan asli."""
    base = (base_name or "Molecule").strip() or "Molecule"
    occupied = set(MOLECULES.keys())
    for idx, entry in enumerate(entries):
        if ignore_index is not None and idx == ignore_index:
            continue
        name = str(entry.get("name", "")).strip()
        if name:
            occupied.add(name)
    if base not in occupied:
        return base

    suffix = _persisted_source_label(source_type)
    candidate = f"{base} [{suffix}]"
    if candidate not in occupied:
        return candidate
    n = 2
    while f"{base} [{suffix} {n}]" in occupied:
        n += 1
    return f"{base} [{suffix} {n}]"
```

A.109 _unique_uploaded_name()

Lokasi dan signature: Baris 3294–3296;
_unique_uploaded_name(base_name, entries)

Fungsi `_unique_uploaded_name` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: _unique_persisted_name. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _unique_uploaded_name(base_name: str, entries):
    # Fungsi v2.2.2 tetap dipertahankan untuk kompatibilitas internal.
    return _unique_persisted_name(base_name, entries, source_type="upload")
```

A.110 _persistent_spec_digest()

> **Lokasi dan signature:** Baris 3299–3311; _persistent_spec_digest(spec, source_type, source_ref, raw_bytes)

Fungsi `_persistent_spec_digest` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 4; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Fingerprint stabil untuk mencegah duplikasi akibat rerun Streamlit.

Interaksi utama yang terdeteksi: dumps, encode, get, hexdigest, sha256, str, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _persistent_spec_digest(spec, source_type: str, source_ref: str = "", raw_bytes: bytes
| None = None):
    """Fingerprint stabil untuk mencegah duplikasi akibat rerun Streamlit."""
    if raw_bytes is not None:
```

```
    return hashlib.sha256(raw_bytes).hexdigest()
payload = {
    "source_type": str(source_type or "custom"),
    "input_name": str(spec.get("name", "")).strip(),
    "smiles": str(spec.get("smiles", "")).strip(),
    "inchi": str(spec.get("inchi", "")).strip(),
    "source_ref": str(source_ref or "").strip(),
}
encoded = json.dumps(payload, ensure_ascii=False, sort_keys=True).encode("utf-8")
return hashlib.sha256(encoded).hexdigest()
```

A.111 persist_molecule_spec()

Lokasi dan signature: Baris 3314–3376; persist_molecule_spec(spec, source_type, source_ref, source_filename, raw_bytes)

Fungsi `persist_molecule_spec` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 5; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Simpan/updates satu molekul ke upload_molekul.json. File JSON v2.2.2 sengaja tetap digunakan agar katalog upload lama langsung terbaca. Entri baru selalu dimasukkan pada indeks 0. Jika nama input yang sama pada jenis sumber yang sama diberi struktur baru, entri lama diperbarui lalu dipindah ke atas.

Interaksi utama yang terdeteksi: _persisted_source_label, _persistent_spec_digest, _unique_persisted_name, enumerate, get, insert, int, load_uploaded_molecules, lower, max, now_text, pop, str, strip, write_json_atomic. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def persist_molecule_spec(spec, source_type: str, source_ref: str = "",
                          source_filename: str = "", raw_bytes: bytes | None = None):
    """
    Simpan/updates satu molekul ke upload_molekul.json.

    File JSON v2.2.2 sengaja tetap digunakan agar katalog upload lama langsung terbaca.
    Entri baru selalu dimasukkan pada indeks 0. Jika nama input yang sama pada jenis
    sumber yang sama diberi struktur baru, entri lama diperbarui lalu dipindah ke atas.
    """
    entries = load_uploaded_molecules()
    source_type = str(source_type or "custom").lower()
    requested_name = str(spec.get("name", "")).strip() or "Molecule"
    digest = _persistent_spec_digest(spec, source_type, source_ref=source_ref,
raw_bytes=raw_bytes)

    # Exact match: rerun Streamlit tidak boleh menulis ulang atau mengubah urutan katalog.
    for entry in entries:
        old_digest = str(entry.get("catalog_digest") or entry.get("sha256") or "")
        if old_digest and old_digest == digest:
            return str(entry.get("name") or requested_name), False

    # Jika user memakai nama input yang sama untuk jenis sumber yang sama, perlakukan
    # sebagai pembaruan molekul, bukan membuat tumpukan duplikat bernama serupa.
    replace_index = None
    for idx, entry in enumerate(entries):
        old_source_type = str(entry.get("source_type") or ("upload" if entry.get("sha256")
else "")).lower()
        old_input_name = str(entry.get("input_name") or entry.get("name") or "").strip()
```

A.112 persist_input_specs()

> **Lokasi dan signature:** Baris 3379–3389; persist_input_specs(specs, source_type)

Fungsi `persist_input_specs` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Persistenkan hasil parser SMILES/InChI/file lokal; kembalikan status perubahan.

Interaksi utama yang terdeteksi: get, persist_molecule_spec, str. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.

- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def persist_input_specs(specs, source_type: str):
    """Persistenkan hasil parser SMILES/InChI/file lokal; kembalikan status perubahan."""
    changed_any = False
    for spec in specs:
        source_ref = str(spec.get("source", ""))
        saved_name, changed = persist_molecule_spec(
            spec, source_type=source_type, source_ref=source_ref
        )
        spec["name"] = saved_name
        changed_any = changed_any or changed
    return specs, changed_any
```

A.113 persist_uploaded_spec()

> **Lokasi dan signature:** Baris 3392–3397; persist_uploaded_spec(spec, source_filename, raw_bytes)

Fungsi `persist_uploaded_spec` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Simpan molekul upload ke upload_molekul.json; molekul baru selalu di indeks 0.

Interaksi utama yang terdeteksi: persist_molecule_spec. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def persist_uploaded_spec(spec, source_filename: str, raw_bytes: bytes):
    """Simpan molekul upload ke upload_molekul.json; molekul baru selalu di indeks 0."""
    return persist_molecule_spec(
        spec, source_type="upload", source_ref=source_filename,
        source_filename=source_filename, raw_bytes=raw_bytes,
    )
```

A.114 build_molecule_catalog()

Lokasi dan signature: Baris 3400–3427; build_molecule_catalog()

Fungsi `build_molecule_catalog` berperan untuk membangun representasi, geometri, visualisasi, atau file keluaran. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Gabungkan semua molekul persisten (terbaru di atas) + tepat 50 preset asli.

Interaksi utama yang terdeteksi: dict, get, int, items, load_uploaded_molecules, lower, str, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def build_molecule_catalog():
```

```
    """Gabungkan semua molekul persisten (terbaru di atas) + tepat 50 preset asli."""
    catalog = {}
    for entry in load_uploaded_molecules():
        name = str(entry.get("name", "")).strip()
        if not name:
            continue
        inferred_type = str(entry.get("source_type") or ("upload" if
entry.get("source_filename") else "custom")).lower()
        catalog[name] = {
            "smiles": entry.get("smiles", ""),
            "inchi": entry.get("inchi", ""),
            "formula": entry.get("formula", ""),
            "charge": int(entry.get("charge", 0)),
            "spin": int(entry.get("spin", 0)),
            "category": entry.get("category", "Persisted molecule"),
            "molblock": entry.get("molblock", ""),
            "uploaded_at": entry.get("uploaded_at", entry.get("saved_at", "")),
            "saved_at": entry.get("saved_at", entry.get("uploaded_at", "")),
            "source_type": inferred_type,
            "source_ref": entry.get("source_ref", ""),
            "source_filename": entry.get("source_filename", ""),
            "persisted_molecule": True,
            # Flag lama tetap ada agar data upload v2.2.2 tetap berperilaku sama.
            "persisted_upload": inferred_type == "upload",
        }
    for name, meta in MOLECULES.items():
```

A.115 _text_input_changed_for_persistence()

Lokasi dan signature: Baris 3430–3441;
_text_input_changed_for_persistence(key, value)

Fungsi `_text_input_changed_for_persistence` berperan untuk menjaga persistensi, serialisasi, dan jejak audit. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 3 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: True hanya sekali untuk isi text-area tertentu sampai isinya berubah lagi.

Interaksi utama yang terdeteksi: encode, get, hexdigest, sha256, str, strip. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def _text_input_changed_for_persistence(key: str, value: str) -> bool:
    """True hanya sekali untuk isi text-area tertentu sampai isinya berubah lagi."""
    normalized = str(value or "").strip()
    state_key = f"_persist_input_digest_{key}"
    if not normalized:
        st.session_state[state_key] = ""
        return False
    digest = hashlib.sha256(normalized.encode("utf-8")).hexdigest()
    if st.session_state.get(state_key) == digest:
        return False
    st.session_state[state_key] = digest
    return True
```

A.116 parse_uploads()

> **Lokasi dan signature:** Baris 3444–3459; parse_uploads(uploaded_files, random_seed, preopt)

Fungsi `parse_uploads` berperan untuk membaca dan menormalkan input struktur molekul. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Path, append, getvalue, make_spec, molecule_from_bytes, persist_uploaded_spec. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.

- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def parse_uploads(uploaded_files, random_seed, preopt):
    specs = []
    errors = []
    catalog_changed = False
    for uf in uploaded_files or []:
        try:
            raw = uf.getvalue()
            mol = molecule_from_bytes(uf.name, raw)
            spec = make_spec(Path(uf.name).stem, mol, f"Upload: {uf.name}", random_seed,
preopt=preopt)
            saved_name, changed = persist_uploaded_spec(spec, uf.name, raw)
            spec["name"] = saved_name
            specs.append(spec)
            catalog_changed = catalog_changed or changed
        except Exception as exc:
            errors.append(f"{uf.name}: {exc}")
    return specs, errors, catalog_changed
```

A.117 unique_names()

Lokasi dan signature: Baris 3462–3470; unique_names(specs)

Fungsi `unique_names` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 1; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: get. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def unique_names(specs):
    used = {}
    for s in specs:
        base = s["name"] or "molecule"
        n = used.get(base, 0) + 1
        used[base] = n
        if n > 1:
            s["name"] = f"{base}-{n}"
    return specs
```

A.118 init_state()

Lokasi dan signature: Baris 3476–3485; init_state()

Fungsi `init_state` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: items. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.

- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def init_state():
    defaults = {
        "worker_thread": None,
        "worker_stop_event": None,
        "current_run_dir": None,
        "last_run_dir": None,
    }
    for k, v in defaults.items():
        if k not in st.session_state:
            st.session_state[k] = v
```

A.119 is_job_running()

Lokasi dan signature: Baris 3488–3490; is_job_running()

Fungsi `is_job_running` berperan untuk mengelola siklus hidup job dan status UI. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 1 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: bool, get, is_alive. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def is_job_running():
    th = st.session_state.get("worker_thread")
    return bool(th is not None and th.is_alive())
```

A.120 start_background_job()

Lokasi dan signature: Baris 3493–3512; start_background_job(specs, config)

Fungsi `start_background_job` berperan untuk mengelola siklus hidup job dan status UI. Jumlah argumen posisi yang terdeteksi adalah 2; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: Event, Thread, get, items, mkdir, now, safe_slug, start, str, strftime, uuid4, write_json_atomic. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def start_background_job(specs, config):
    project_slug = safe_slug(config.get("project_name", "QM_Project"))
    run_id = project_slug + "__" + datetime.now().strftime("%Y%m%d_%H%M%S") + "_" +
uuid.uuid4().hex[:8]
    run_dir = RUNS_DIR / run_id
    run_dir.mkdir(parents=True, exist_ok=True)
    write_json_atomic(run_dir / "config.json", config)
    manifest = [{k: v for k, v in s.items() if k != "molblock"} for s in specs]
    write_json_atomic(run_dir / "input_manifest.json", manifest)
    stop_event = threading.Event()
    th = threading.Thread(
        target=job_worker,
        args=(str(run_dir), specs, config, stop_event),
        daemon=True,
        name=f"qm-job-{run_id}",
    )
    st.session_state.worker_stop_event = stop_event
    st.session_state.worker_thread = th
    st.session_state.current_run_dir = str(run_dir)
    st.session_state.last_run_dir = str(run_dir)
    th.start()
```

A.121 request_stop()

Lokasi dan signature: Baris 3515–3518; request_stop()

Fungsi `request_stop` berperan untuk mengelola siklus hidup job dan status UI. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: get, set. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.

- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def request_stop():
    ev = st.session_state.get("worker_stop_event")
    if ev is not None:
        ev.set()
```

A.122 copy_text_component()

> **Lokasi dan signature:** Baris 3521–3562; copy_text_component(text, label, key)

Fungsi `copy_text_component` berperan untuk mendukung orkestrasi, kompatibilitas, atau utilitas internal aplikasi. Jumlah argumen posisi yang terdeteksi adalah 3; tubuh fungsi memiliki 0 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Docstring sumber: Tombol salin eksplisit dengan Clipboard API + fallback execCommand.

Interaksi utama yang terdeteksi: abs, dumps, hash, html, safe_slug, str. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def copy_text_component(text, label="📋 Salin", key="copy"):
    """Tombol salin eksplisit dengan Clipboard API + fallback execCommand."""
    value_js = json.dumps(str(text), ensure_ascii=False)
```

```
    element_id = safe_slug(f"copy_{key}_{abs(hash(str(text))) % 10000000}")
    html = f"""
    <div style="display:flex;justify-content:flex-end;align-items:center;height:38px;">
      <button id="{element_id}" style="padding:7px 12px;border:1px solid #999;border-
radius:7px;cursor:pointer;background:#fff;font-weight:600;">
        {label}
      </button>
      <span id="{element_id}_msg" style="margin-left:8px;font-family:sans-serif;font-
size:12px;"></span>
    </div>
    <script>
    const textToCopy = {value_js};
    const btn = document.getElementById('{element_id}');
    const msg = document.getElementById('{element_id}_msg');
    btn.addEventListener('click', async () => {{
      let ok = false;
      try {{
        if (navigator.clipboard && window.isSecureContext) {{
          await navigator.clipboard.writeText(textToCopy);
          ok = true;
        }}
      }} catch (e) {{}}
      if (!ok) {{
        try {{
          const ta = document.createElement('textarea');
```

A.123 render_monitor_and_results()

Lokasi dan signature: Baris 4079–4387; render_monitor_and_results()

Fungsi `render_monitor_and_results` berperan untuk mengelola siklus hidup job dan status UI. Jumlah argumen posisi yang terdeteksi adalah 0; tubuh fungsi memiliki 2 pernyataan return dan 0 pernyataan raise. Informasi ini membantu menentukan apakah fungsi hanya mentransformasi data, menjalankan efek samping, atau dapat menghentikan alur melalui exception.

Kode sumber tidak memberikan docstring eksplisit untuk fungsi ini. Oleh karena itu, perilaku harus diverifikasi melalui argumen, pemanggilan internal, nilai return, perubahan file/state, dan uji pada input kecil. Dokumentasi tambahan pada versi berikutnya sebaiknya menyebutkan tipe data, satuan, efek samping, exception, dan contoh minimal.

Interaksi utama yang terdeteksi: DataFrame, MolFromMolBlock, MolToImage, Path, RemoveHs, append, caption, code, columns, copy_text_component, dataframe, download_button, encode, enumerate, error, exists, expander, get. Daftar ini bersifat struktural dan dapat mencakup method library, bukan hanya fungsi lokal. Saat melakukan refactor, pengembang perlu memeriksa kontrak setiap interaksi: tipe objek, mutabilitas, path, satuan, dan penanganan kegagalan.

Kontrak verifikasi

- Uji input valid minimum dan cocokkan keluaran dengan tujuan fungsi.
- Uji input kosong, format salah, atau backend tidak tersedia; pastikan pesan error informatif.
- Periksa apakah fungsi menulis file, mengubah session state, menjalankan subprocess, atau bergantung pada seed.
- Untuk nilai numerik, periksa satuan, bentuk array, NaN/inf, toleransi, dan ketergantungan versi library.
- Tambahkan regression test sebelum mengubah parser, konversi energi, atau struktur hasil.

```
def render_monitor_and_results():
    run_dir_text = st.session_state.get("current_run_dir") or
st.session_state.get("last_run_dir")
    if not run_dir_text:
        st.subheader("Apa yang dapat dihitung?")
        st.markdown(
            """
1. Input struktur dari preset, **SMILES, InChI, file lokal, atau upload** berbagai format
kimia.
2. Geometri awal 3D dengan RDKit; pre-optimasi MMFF94/UFF dapat dimatikan.
3. Pilihan **HF (Auto/RHF/ROHF/UHF), DFT, MP2 single-point, semiempirik PySCF, MOPAC,
xTB**, dan backend ORCA eksternal.
   PySCF semiempirik mempertahankan AM1/MINDO/3; MOPAC menambah MNDO, PM3, RM1, PM6, PM7
dan koreksinya.
4. Basis set >20 pilihan dari small hingga very large; default **def2-SVP**.
5. Job **Single Point**, **Gradient**, **Geometry Optimization**, **Frequency Analysis**,
dan **Geometry Optimization + Frequency**.
6. Optimizer **geomeTRIC**, **PyBerny**, atau ASE (opsional) untuk PySCF; backend
eksternal memakai optimizer internal.
7. Energi dalam **Hartree/Eh (a.u.), eV, kJ/mol, kcal/mol, cm⁻¹** dan frekuensi vibrasi
dalam **cm⁻¹**.
8. Output final **XYZ + MOL + SDF + PDB**; MOL/SDF/PDB mempertahankan konektivitas/ikatan
dari struktur input.
9. `progress.log`, `summary.log`, `config.json`, manifest input, checkpoint PySCF, CSV
koordinat/orbital/Mulliken/frekuensi.
10. Nama proyek membedakan folder dan salinan nama file input/output; worker tetap
berjalan terpisah dan tombol Stop tersedia.
            """
        )
        return

    run_dir = Path(run_dir_text)
    status = read_json_safe(run_dir / "status.json", {}) or {}
    payload = read_json_safe(run_dir / "results.json", {}) or {}
    results = payload.get("results", [])
    errors = payload.get("errors", [])
```

LAMPIRAN B — KODE SUMBER LENGKAP V2.3.4

Lampiran ini mempertahankan isi kode Python terlampir secara lengkap dengan nomor baris untuk memudahkan audit silang. Tipografi monospace dan ukuran lebih kecil digunakan khusus untuk kode; isi sintaks tidak diubah. Apabila kode diperbarui, dokumentasi dan pengujian harus diperbarui bersama-sama.

```
0001  import hashlib
0002  import importlib
0003  import io
0004  import inspect
0005  import json
0006  import math
0007  import os
0008  import platform
0009  import re
0010  import shutil
0011  import subprocess
0012  import sys
0013  import tempfile
0014  import threading
0015  import time
0016  import traceback
0017  import types
0018  import uuid
0019  from datetime import datetime
0020  from pathlib import Path
0021
0022
0023  # ============================================================
0024  # BOOTSTRAP DEPENDENSI PYTHON (v2.3.4)
0025  # ============================================================
0026  def ensure_python_package(import_name: str, pip_name: str | None = None):
0027      """Import paket; bila belum ada, instal otomatis dengan interpreter aktif.
0028
0029      Nama paket pip hanya berasal dari pemetaan internal kode ini (bukan input user),
0030      sehingga instalasi otomatis tetap terkontrol. Untuk executable eksternal seperti
0031      ORCA/MOPAC/xTB/Open Babel, instalasi sistem tetap tidak dapat dipaksakan via
pip.
0032      """
0033      try:
0034          return importlib.import_module(import_name)
0035      except ModuleNotFoundError:
0036          package = pip_name or import_name.split(".", 1)[0]
0037          cmd = [
0038              sys.executable, "-m", "pip", "install",
0039              "--disable-pip-version-check", "--prefer-binary", package,
0040          ]
0041          print(f"[AUTO-INSTALL] Dependensi '{import_name}' belum tersedia.
Menjalankan: {' '.join(cmd)}")
0042          try:
0043              subprocess.check_call(cmd)
0044          except Exception as exc:
0045              raise RuntimeError(
0046                  f"Dependensi Python '{import_name}' belum tersedia dan instalasi
otomatis gagal. "
0047                  f"Perintah yang dicoba: {' '.join(cmd)}. Detail: {exc}"
0048              ) from exc
0049          importlib.invalidate_caches()
0050          try:
```

```
0051              return importlib.import_module(import_name)
0052          except Exception as exc:
0053              raise RuntimeError(
0054                  f"Paket '{package}' selesai dicoba instal, tetapi modul
'{import_name}' masih tidak dapat diimpor: {exc}"
0055              ) from exc
0056
0057
0058  # Dependensi inti harus tersedia sebelum import utama agar aplikasi tidak langsung
0059  # berhenti hanya karena satu library Python belum terpasang.
0060  ensure_python_package("numpy", "numpy")
0061  ensure_python_package("pandas", "pandas")
0062  ensure_python_package("streamlit", "streamlit")
0063  ensure_python_package("rdkit", "rdkit")
0064  ensure_python_package("pyscf", "pyscf")
0065
0066  import numpy as np
0067  import pandas as pd
0068  import streamlit as st
0069  import streamlit.components.v1 as components
0070  from rdkit import Chem
0071  from rdkit.Chem import AllChem, Draw, Descriptors, rdMolDescriptors
0072
0073  try:
0074      from rdkit.Chem import rdDetermineBonds
0075  except Exception:
0076      rdDetermineBonds = None
0077
0078  from pyscf import dft, gto, lib, mp, scf
0079
0080  # ============================================================
0081  # KONFIGURASI APLIKASI
0082  # ============================================================
0083  st.set_page_config(
0084      page_title="Komputasi Molekul QM | PySCF / xTB / ORCA",
0085      page_icon="⚛",
0086      layout="wide",
0087  )
0088
0089  # Versi aplikasi: 2.3.4 — seluruh fitur v2.2.3 dipertahankan dan ditambah:
0090  # semua penambahan molekul (SMILES, InChI, file lokal, dan upload) disimpan permanen
0091  # pada katalog JSON; molekul yang baru ditambahkan ditempatkan di posisi paling atas
0092  # sebagai molekul tersimpan/bawaan pengguna dan tetap tersedia setelah aplikasi
restart.
0093  # Fitur lama tetap mencakup default preset kosong, separator Nama=SMILES /
Nama=InChI,
0094  # katalog upload persisten, auto-install dependensi Python, project naming,
frequency jobs,
0095  # GPU detection, 50 preset, MOPAC semiempirical, HF reference selection, folder
opener
0096  # WSL/Windows, dan tombol salin log.
0097  # Konstanta konversi energi
0098  HARTREE_TO_EV = 27.211386245988
0099  HARTREE_TO_KJ_MOL = 2625.4996394799
0100  HARTREE_TO_KCAL_MOL = 627.5094740631
0101  HARTREE_TO_CM1 = 219474.6313705
0102
0103  APP_DIR = Path(__file__).resolve().parent
0104  RUNS_DIR = APP_DIR / "qm_runs"
0105  RUNS_DIR.mkdir(parents=True, exist_ok=True)
0106  UPLOAD_MOLECULES_FILE = APP_DIR / "upload_molekul.json"
0107
0108  # Molekul bawaan lama dipertahankan agar fitur yang sudah ada tetap tersedia.
```

```
MOLECULES = {
    "Benzene": {
        "smiles": "c1ccccc1",
        "formula": "C6H6",
        "charge": 0,
        "spin": 0,
    },
    "Toluene": {
        "smiles": "Cc1ccccc1",
        "formula": "C7H8",
        "charge": 0,
        "spin": 0,
    },
    "Phenol": {
        "smiles": "Oc1ccccc1",
        "formula": "C6H6O",
        "charge": 0,
        "spin": 0,
    },
}

# Preset tambahan. Tiga preset lama di atas tidak diubah; daftar diperluas menjadi
# tepat 50 molekul, dari sistem sangat kecil sampai obat antikanker berukuran besar.
# Formula dihitung kembali oleh RDKit saat input dibuat; field formula di sini hanya
# metadata ringkas untuk preset lama/kompatibilitas.
MOLECULES.update({
    "Water": {"smiles": "O", "formula": "H2O", "charge": 0, "spin": 0, "category":
"Small molecule"},
    "Methane": {"smiles": "C", "formula": "CH4", "charge": 0, "spin": 0, "category":
"Small molecule"},
    "Ethane": {"smiles": "CC", "formula": "C2H6", "charge": 0, "spin": 0,
"category": "Small molecule"},
    "Ammonia": {"smiles": "N", "formula": "NH3", "charge": 0, "spin": 0, "category":
"Small molecule"},
    "Methanol": {"smiles": "CO", "formula": "CH4O", "charge": 0, "spin": 0,
"category": "Small molecule"},
    "Ethanol": {"smiles": "CCO", "formula": "C2H6O", "charge": 0, "spin": 0,
"category": "Small molecule"},
    "Acetone": {"smiles": "CC(=O)C", "formula": "C3H6O", "charge": 0, "spin": 0,
"category": "Small molecule"},
    "Acetic acid": {"smiles": "CC(=O)O", "formula": "C2H4O2", "charge": 0, "spin":
0, "category": "Small molecule"},
    "Formaldehyde": {"smiles": "C=O", "formula": "CH2O", "charge": 0, "spin": 0,
"category": "Small molecule"},
    "Urea": {"smiles": "NC(N)=O", "formula": "CH4N2O", "charge": 0, "spin": 0,
"category": "Small molecule"},
    "Acetonitrile": {"smiles": "CC#N", "formula": "C2H3N", "charge": 0, "spin": 0,
"category": "Small molecule"},
    "Nitrobenzene": {"smiles": "[O-][N+](=O)c1ccccc1", "formula": "C6H5NO2",
"charge": 0, "spin": 0, "category": "Aromatic"},
    "Aniline": {"smiles": "Nc1ccccc1", "formula": "C6H7N", "charge": 0, "spin": 0,
"category": "Aromatic"},
    "Pyridine": {"smiles": "n1ccccc1", "formula": "C5H5N", "charge": 0, "spin": 0,
"category": "Heteroaromatic"},
    "Naphthalene": {"smiles": "c1ccc2ccccc2c1", "formula": "C10H8", "charge": 0,
"spin": 0, "category": "Aromatic"},
    "Caffeine": {"smiles": "Cn1c(=O)c2c(ncn2C)n(C)c1=O", "formula": "C8H10N4O2",
"charge": 0, "spin": 0, "category": "Bioactive"},
    "Paracetamol": {"smiles": "CC(=O)NC1=CC=C(O)C=C1", "formula": "C8H9NO2",
"charge": 0, "spin": 0, "category": "Drug"},
    "Aspirin": {"smiles": "CC(=O)Oc1ccccc1C(=O)O", "formula": "C9H8O4", "charge": 0,
"spin": 0, "category": "Drug"},
    "Ibuprofen": {"smiles": "CC(C)Cc1ccc(cc1)[C@@H](C)C(=O)O", "formula":
```

```
"C13H18O2", "charge": 0, "spin": 0, "category": "Drug"},
0154      "Naproxen": {"smiles": "COc1ccc2cc([C@@H](C)C(=O)O)ccc2c1", "formula":
"C14H14O3", "charge": 0, "spin": 0, "category": "Drug"},
0155      "Glucose": {"smiles": "OC[C@H]1O[C@@H](O)[C@H](O)[C@@H](O)[C@@H]1O", "formula":
"C6H12O6", "charge": 0, "spin": 0, "category": "Biomolecule"},
0156      "Curcumin": {"smiles": "COc1cc(/C=C/C(=O)CC(=O)/C=C/c2ccc(O)c(OC)c2)ccc1O",
"formula": "C21H20O6", "charge": 0, "spin": 0, "category": "Natural product"},
0157      "Quercetin": {"smiles": "O=c1c(O)c(-c2ccc(O)c(O)c2)oc2cc(O)cc(O)c12", "formula":
"C15H10O7", "charge": 0, "spin": 0, "category": "Natural product"},
0158      "Resveratrol": {"smiles": "Oc1ccc(/C=C/c2cc(O)cc(O)c2)cc1", "formula":
"C14H12O3", "charge": 0, "spin": 0, "category": "Natural product"},
0159      "Tamoxifen": {"smiles": "CC/C(=C(\\c1ccccc1)c2ccc(OCCN(C)C)cc2)/c3ccccc3",
"formula": "C26H29NO", "charge": 0, "spin": 0, "category": "Breast cancer"},
0160      "Letrozole": {"smiles": "N#Cc1ccc(cc1)C(c2ccc(C#N)cc2)n3cncn3", "formula":
"C17H11N5", "charge": 0, "spin": 0, "category": "Breast cancer"},
0161      "Anastrozole": {"smiles": "CC(C)(C#N)c1cc(cc(c1)C(C)(C)C#N)Cn2cncn2", "formula":
"C17H19N5", "charge": 0, "spin": 0, "category": "Breast cancer"},
0162      "Exemestane": {"smiles":
"C[C@]12CC[C@H]3[C@H]([C@@H]1CCC2=O)CC(=C)C4=CC(=O)C=C[C@]34C", "formula": "C20H24O2",
"charge": 0, "spin": 0, "category": "Breast cancer"},
0163      "Methotrexate": {"smiles":
"CN(CC1=CN=C2C(=N1)C(=NC(=N2)N)N)C3=CC=C(C=C3)C(=O)N[C@@H](CCC(=O)O)C(=O)O", "formula":
"C20H22N8O5", "charge": 0, "spin": 0, "category": "Anticancer"},
0164      "Cyclophosphamide": {"smiles": "O=P1(N(CCCl)CCCl)OCCCN1", "formula":
"C7H15Cl2N2O2P", "charge": 0, "spin": 0, "category": "Anticancer"},
0165      "5-Fluorouracil": {"smiles": "O=c1[nH]cc(F)c(=O)[nH]1", "formula": "C4H3FN2O2",
"charge": 0, "spin": 0, "category": "Anticancer / skin"},
0166      "Lapatinib": {"smiles":
"CS(=O)(=O)CCNCC1=CC=C(O1)C2=CC3=C(C=C2)N=CN=C3NC4=CC(=C(C=C4)OCC5=CC(=CC=C5)F)Cl",
"formula": "C29H26ClFN4O4S", "charge": 0, "spin": 0, "category": "Breast cancer"},
0167      "Olaparib": {"smiles":
"C1CC1C(=O)N2CCN(CC2)C(=O)C3=C(C=CC(=C3)CC4=NNC(=O)C5=CC=CC=C54)F", "formula":
"C24H23FN4O3", "charge": 0, "spin": 0, "category": "Breast / ovarian cancer"},
0168      "Sunitinib": {"smiles":
"CCN(CC)CCNC(=O)C1=C(NC(=C1C)/C=C\\2/C3=C(C=CC(=C3)F)NC2=O)C", "formula": "C22H27FN4O2",
"charge": 0, "spin": 0, "category": "Kidney cancer"},
0169      "Sorafenib": {"smiles":
"CNC(=O)c1cc(Oc2ccc(NC(=O)Nc3ccc(Cl)c(C(F)(F)F)c3)cc2)ccn1", "formula": "C21H16ClF3N4O3",
"charge": 0, "spin": 0, "category": "Kidney cancer"},
0170      "Axitinib": {"smiles":
"CNC(=O)C1=CC=CC=C1SC2=CC3=C(C=C2)C(=NN3)/C=C/C4=CC=CC=N4", "formula": "C22H18N4OS",
"charge": 0, "spin": 0, "category": "Kidney cancer"},
0171      "Cabozantinib": {"smiles":
"COC1=CC2=C(C=CN=C2C=C1OC)OC3=CC=C(C=C3)NC(=O)C4(CC4)C(=O)NC5=CC=C(C=C5)F", "formula":
"C28H24FN3O5", "charge": 0, "spin": 0, "category": "Kidney cancer / medium-large"},
0172      "Vismodegib": {"smiles":
"CS(=O)(=O)C1=CC(=C(C=C1)C(=O)NC2=CC(=C(C=C2)Cl)C3=CC=CC=N3)Cl", "formula":
"C19H14Cl2N2O3S", "charge": 0, "spin": 0, "category": "Skin cancer"},
0173      "Vemurafenib": {"smiles":
"CCCS(=O)(=O)NC1=C(C(=C(C=C1)F)C(=O)C2=CNC3=C2C=C(C=N3)C4=CC=C(C=C4)Cl)F", "formula":
"C23H18ClF2N3O3S", "charge": 0, "spin": 0, "category": "Melanoma / skin cancer"},
0174      "Dabrafenib": {"smiles":
"CC(C)(C)C1=NC(=C(S1)C2=NC(=NC=C2)N)C3=C(C(=CC=C3)NS(=O)(=O)C4=C(C=CC=C4F)F)F", "formula":
"C23H20F3N5O2S2", "charge": 0, "spin": 0, "category": "Melanoma / skin cancer"},
0175      "Dacarbazine": {"smiles": "CN(C)N=Nc1[nH]cnc1C(N)=O", "formula": "C6H10N6O",
"charge": 0, "spin": 0, "category": "Melanoma / anticancer"},
0176      "Temozolomide": {"smiles": "CN1C(=O)N2C=NC(=C2N=N1)C(N)=O", "formula":
"C6H6N6O2", "charge": 0, "spin": 0, "category": "Anticancer"},
0177      "Erlotinib": {"smiles": "COCCOc1cc2ncnc(Nc3cccc(C#C)c3)c2cc1OCCOC", "formula":
"C22H23N3O4", "charge": 0, "spin": 0, "category": "Anticancer"},
0178      "Gefitinib": {"smiles": "COc1cc2ncnc(Nc3ccc(F)c(Cl)c3)c2cc1OCCCN1CCOCC1",
"formula": "C22H24ClFN4O3", "charge": 0, "spin": 0, "category": "Anticancer"},
0179      "Imatinib": {"smiles":
```

```
"Cc1ccc(cc1Nc2nccc(n2)c3cccnc3)NC(=O)c4ccc(CN5CCN(C)CC5)cc4", "formula": "C29H31N7O",
"charge": 0, "spin": 0, "category": "Anticancer / large"},
0180      "Hydroxyurea": {"smiles": "NC(=O)NO", "formula": "CH4N2O2", "charge": 0, "spin":
0, "category": "Anticancer"},
0181      "Estradiol": {"smiles": "CC12CCC3C(C1CCC2O)CCC4=CC(O)=CC=C34", "formula":
"C18H24O2", "charge": 0, "spin": 0, "category": "Hormone / breast-cancer reference"},
0182  })
0183
0184  assert len(MOLECULES) == 50, f"Preset molecule count must be 50, got
{len(MOLECULES)}"
0185
0186  # Lebih dari 20 basis set, dikelompokkan berdasarkan ukuran praktis.
0187  BASIS_GROUPS = {
0188      "Small": [
0189          "STO-3G",
0190          "3-21G",
0191          "6-31G",
0192          "6-31G*",
0193          "6-31G**",
0194          "def2-SV(P)",
0195          "def2-SVP",
0196          "cc-pVDZ",
0197          "pcseg-0",
0198      ],
0199      "Medium": [
0200          "6-311G",
0201          "6-311G*",
0202          "6-311G**",
0203          "def2-SVPD",
0204          "def2-TZVP",
0205          "def2-TZVPP",
0206          "cc-pVTZ",
0207          "aug-cc-pVDZ",
0208          "pcseg-1",
0209      ],
0210      "Large": [
0211          "def2-TZVPD",
0212          "def2-TZVPPD",
0213          "def2-QZVP",
0214          "cc-pVQZ",
0215          "aug-cc-pVTZ",
0216          "pcseg-2",
0217      ],
0218      "Very large": [
0219          "def2-QZVPP",
0220          "def2-QZVPD",
0221          "def2-QZVPPD",
0222          "aug-cc-pVQZ",
0223          "pcseg-3",
0224      ],
0225  }
0226  BASIS_SETS = [b for group in BASIS_GROUPS.values() for b in group]
0227
0228  PYSCF_METHODS = {
0229      "HF": {"family": "HF", "xc": None, "opt": True, "hf_reference": "AUTO"},
0230      "RHF": {"family": "HF", "xc": None, "opt": True, "hf_reference": "RHF"},
0231      "ROHF": {"family": "HF", "xc": None, "opt": True, "hf_reference": "ROHF"},
0232      "UHF": {"family": "HF", "xc": None, "opt": True, "hf_reference": "UHF"},
0233      "LDA (SVWN)": {"family": "DFT", "xc": "SVWN", "opt": True},
0234      "PBE": {"family": "DFT", "xc": "PBE", "opt": True},
0235      "BP86": {"family": "DFT", "xc": "BP86", "opt": True},
0236      "BLYP": {"family": "DFT", "xc": "BLYP", "opt": True},
0237      "B3LYP": {"family": "DFT", "xc": "B3LYP", "opt": True},
```

```
0238      "B3LYP-D3BJ": {"family": "DFT", "xc": "B3LYP-D3BJ", "opt": True},
0239      "PBE0": {"family": "DFT", "xc": "PBE0", "opt": True},
0240      "PBE0-D3BJ": {"family": "DFT", "xc": "PBE0-D3BJ", "opt": True},
0241      "TPSS": {"family": "DFT", "xc": "TPSS", "opt": True},
0242      "TPSSh": {"family": "DFT", "xc": "TPSSh", "opt": True},
0243      "M06-L": {"family": "DFT", "xc": "M06-L", "opt": True},
0244      "M06-2X": {"family": "DFT", "xc": "M06-2X", "opt": True},
0245      "CAM-B3LYP": {"family": "DFT", "xc": "CAM-B3LYP", "opt": True},
0246      "ωB97X-D3BJ": {"family": "DFT", "xc": "WB97X-D3BJ", "opt": True},
0247      "SCAN": {"family": "DFT", "xc": "SCAN", "opt": True},
0248      "MP2": {"family": "MP2", "xc": None, "opt": False},
0249  }
0250
0251  SEMIEMPIRICAL_METHODS = ["MINDO/3", "AM1"]
0252  # MOPAC menyediakan keluarga semiempirik yang jauh lebih luas. PM4/PM5 sengaja
0253  # tidak dibuat sebagai alias palsu karena bukan Hamiltonian standar MOPAC.
0254  MOPAC_METHODS = [
0255      "MNDO", "AM1", "PM3", "RM1", "MNDOD",
0256      "PM6", "PM6-D3", "PM6-DH+", "PM6-DH2", "PM6-D3H4",
0257      "PM6-DH2X", "PM6-D3H4X", "PM7",
0258  ]
0259  XTB_METHODS = ["GFN2-xTB", "GFN1-xTB", "GFN0-xTB"]
0260  ORCA_METHODS = ["HF", "RHF", "ROHF", "UHF", "B3LYP", "PBE", "PBE0", "BP86", "M06-
2X", "CAM-B3LYP"]
0261
0262  JOBS = [
0263      "Single Point",
0264      "Single Point + Gradient",
0265      "Geometry Optimization",
0266      "Geometry Optimization + Final Gradient",
0267      "Frequency Analysis",
0268      "Geometry Optimization + Frequency",
0269  ]
0270
0271
0272  # Add-Ons v2.3.4. None mempertahankan perilaku komputasi lama.
0273  ADDON_OPTIONS = [
0274      "None",
0275      "IR",
0276      "Thermodynamics",
0277      "UV-Vis",
0278      "NMR",
0279      "Raman",
0280      "ΔH f (g3mp2)",
0281      "NBO (def2-tzpp)",
0282  ]
0283
0284  # Konstanta tambahan untuk spektrum/deskriptor.
0285  EV_NM = 1239.8419843320026
0286  DEFAULT_TEMPERATURE_K = 298.15
0287  DEFAULT_PRESSURE_PA = 101325.0
0288
0289  OPTIMIZERS = [
0290      "geomeTRIC melalui PySCF",
0291      "PyBerny melalui PySCF",
0292      "ASE-BFGS melalui PySCF (opsional)",
0293  ]
0294
0295  # ============================================================
0296  # UTILITAS UMUM
0297  # ============================================================
0298  class StopRequested(RuntimeError):
0299      pass
```

```
0300
0301
0302  def safe_slug(text: str) -> str:
0303      text = re.sub(r"[^A-Za-z0-9._-]+", "_", str(text).strip())
0304      return text.strip("._-") or "molecule"
0305
0306
0307  def now_text() -> str:
0308      return datetime.now().strftime("%Y-%m-%d %H:%M:%S")
0309
0310
0311  def write_json_atomic(path: Path, data):
0312      path = Path(path)
0313      tmp = path.with_suffix(path.suffix + ".tmp")
0314      with open(tmp, "w", encoding="utf-8") as f:
0315          json.dump(data, f, ensure_ascii=False, indent=2, allow_nan=True)
0316      os.replace(tmp, path)
0317
0318
0319  def read_json_safe(path: Path, default=None):
0320      try:
0321          with open(path, "r", encoding="utf-8") as f:
0322              return json.load(f)
0323      except Exception:
0324          return default
0325
0326
0327  def append_log(path: Path, message: str):
0328      path.parent.mkdir(parents=True, exist_ok=True)
0329      with open(path, "a", encoding="utf-8") as f:
0330          f.write(f"[{now_text()}] {message}\n")
0331          f.flush()
0332
0333
0334  def is_wsl_environment():
0335      """Deteksi WSL tanpa menambah dependensi eksternal."""
0336      try:
0337          if os.environ.get("WSL_DISTRO_NAME") or os.environ.get("WSL_INTEROP"):
0338              return True
0339          return "microsoft" in Path("/proc/version").read_text(encoding="utf-8",
errors="ignore").lower()
0340      except Exception:
0341          return False
0342
0343
0344  def open_folder_os(folder: str):
0345      """Buka folder pada OS host. Khusus WSL gunakan explorer.exe + wslpath -w."""
0346      resolved = Path(folder).expanduser().resolve()
0347      p = str(resolved)
0348      try:
0349          system = platform.system().lower()
0350          if is_wsl_environment():
0351              cp = subprocess.run(
0352                  ["wslpath", "-w", p],
0353                  stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True,
timeout=10,
0354              )
0355              win_path = cp.stdout.strip() if cp.returncode == 0 else p
0356              explorer = shutil.which("explorer.exe") or "/mnt/c/Windows/explorer.exe"
0357              subprocess.Popen([explorer, win_path], stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
0358              return True, win_path
0359          if system.startswith("windows"):
```

```
0360              os.startfile(p)  # type: ignore[attr-defined]
0361              return True, p
0362          if system == "darwin":
0363              subprocess.Popen(["open", p])
0364              return True, p
0365          opener = shutil.which("xdg-open")
0366          if opener:
0367              subprocess.Popen([opener, p], stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
0368              return True, p
0369          return False, "xdg-open tidak ditemukan"
0370      except Exception as exc:
0371          return False, str(exc)
0372
0373
0374  def find_mopac_executable():
0375      """Cari executable MOPAC yang umum di Linux/WSL/Windows PATH."""
0376      for name in ("mopac", "MOPAC", "mopac.exe", "MOPAC2016.exe", "MOPAC2023.exe",
"MOPAC2024.exe"):
0377          found = shutil.which(name)
0378          if found:
0379              return found
0380      return None
0381
0382
0383  def detect_gpu_info():
0384      """Deteksi CUDA/NVIDIA secara ringan; tidak memaksa PyTorch/CuPy terpasang."""
0385      info = {
0386          "detected": False,
0387          "name": "",
0388          "count": 0,
0389          "cuda_visible_devices": os.environ.get("CUDA_VISIBLE_DEVICES", ""),
0390          "gpu4pyscf": False,
0391      }
0392      nvidia_smi = shutil.which("nvidia-smi")
0393      if nvidia_smi:
0394          try:
0395              cp = subprocess.run(
0396                  [nvidia_smi, "--query-gpu=name", "--format=csv,noheader"],
0397                  stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True,
timeout=10,
0398              )
0399              names = [x.strip() for x in cp.stdout.splitlines() if x.strip()]
0400              if cp.returncode == 0 and names:
0401                  info.update(detected=True, name=", ".join(names), count=len(names))
0402          except Exception:
0403              pass
0404      try:
0405          import gpu4pyscf  # noqa: F401
0406          info["gpu4pyscf"] = True
0407      except Exception:
0408          pass
0409      return info
0410
0411
0412  def energy_units(e_h):
0413      if e_h is None:
0414          return {
0415              "hartree": None,
0416              "au": None,
0417              "ev": None,
0418              "kj_mol": None,
0419              "kcal_mol": None,
```

```
            "cm-1": None,
        }
    e = float(e_h)
    return {
        "hartree": e,
        "au": e,
        "ev": e * HARTREE_TO_EV,
        "kj_mol": e * HARTREE_TO_KJ_MOL,
        "kcal_mol": e * HARTREE_TO_KCAL_MOL,
        "cm-1": e * HARTREE_TO_CM1,
    }


def backend_status():
    semi_ok = False
    try:
        from pyscf import semiempirical  # noqa: F401
        semi_ok = True
    except Exception:
        pass
    gpu = detect_gpu_info()
    return {
        "PySCF": True,
        "PySCF Semiempirical": semi_ok,
        "MOPAC CLI": find_mopac_executable() is not None,
        "xTB CLI": shutil.which("xtb") is not None,
        "ORCA CLI": shutil.which("orca") is not None,
        "Open Babel": shutil.which("obabel") is not None,
        "GPU/CUDA": bool(gpu["detected"]),
        "GPU4PySCF": bool(gpu["gpu4pyscf"]),
    }


# ============================================================
# PEMBACAAN DAN NORMALISASI STRUKTUR
# ============================================================
def build_rdkit_3d(smiles: str, random_seed: int = 20260820):
    """Fungsi lama dipertahankan: bangun struktur 3D dari SMILES."""
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        raise ValueError(f"SMILES tidak valid: {smiles}")
    return prepare_molecule_3d(mol, random_seed=random_seed, preopt=True)


def prepare_molecule_3d(mol, random_seed=20260820, preopt=True):
    if mol is None:
        raise ValueError("Objek molekul kosong/tidak valid.")

    mol = Chem.Mol(mol)
    try:
        Chem.SanitizeMol(mol)
    except Exception:
        # Beberapa struktur logam/koordinasi dapat gagal sanitasi; tetap dicoba.
        pass

    mol = Chem.AddHs(mol, addCoords=True)
    need_embed = mol.GetNumConformers() == 0
    if not need_embed:
        try:
            need_embed = not bool(mol.GetConformer().Is3D())
        except Exception:
            need_embed = True

```

```
0483      if need_embed:
0484          params = AllChem.ETKDGv3()
0485          params.randomSeed = int(random_seed)
0486          params.useRandomCoords = True
0487          status = AllChem.EmbedMolecule(mol, params)
0488          if status != 0:
0489              params.useRandomCoords = True
0490              status = AllChem.EmbedMolecule(mol, params)
0491          if status != 0:
0492              raise RuntimeError("RDKit gagal membuat conformer 3D awal.")
0493
0494      if preopt:
0495          try:
0496              if AllChem.MMFFHasAllMoleculeParams(mol):
0497                  AllChem.MMFFOptimizeMolecule(mol, mmffVariant="MMFF94",
maxIters=500)
0498              else:
0499                  AllChem.UFFOptimizeMolecule(mol, maxIters=500)
0500          except Exception:
0501              pass
0502      return mol
0503
0504
0505  def infer_bonds_xyz(mol, charge=0):
0506      if mol is None or rdDetermineBonds is None:
0507          return mol
0508      try:
0509          rdDetermineBonds.DetermineBonds(mol, charge=int(charge))
0510      except Exception:
0511          try:
0512              rdDetermineBonds.DetermineConnectivity(mol)
0513          except Exception:
0514              pass
0515      return mol
0516
0517
0518  def mol_from_openbabel_bytes(filename: str, data: bytes):
0519      obabel = shutil.which("obabel")
0520      if not obabel:
0521          raise ValueError(
0522              f"Format {Path(filename).suffix or '(tanpa ekstensi)'} belum didukung
langsung oleh RDKit. "
0523              "Pasang Open Babel agar format tambahan dapat dikonversi otomatis."
0524          )
0525      suffix = Path(filename).suffix or ".dat"
0526      with tempfile.TemporaryDirectory() as td:
0527          inp = Path(td) / ("input" + suffix)
0528          out = Path(td) / "converted.sdf"
0529          inp.write_bytes(data)
0530          cp = subprocess.run(
0531              [obabel, str(inp), "-O", str(out)],
0532              stdout=subprocess.PIPE,
0533              stderr=subprocess.PIPE,
0534              text=True,
0535              timeout=120,
0536          )
0537          if cp.returncode != 0 or not out.exists():
0538              raise ValueError(f"Open Babel gagal membaca {filename}:
{cp.stderr.strip()}")
0539          supplier = Chem.SDMolSupplier(str(out), removeHs=False)
0540          mol = next((m for m in supplier if m is not None), None)
0541          if mol is None:
0542              raise ValueError(f"Open Babel menghasilkan SDF, tetapi RDKit gagal
```

```
membacanya: {filename}")
0543          return mol
0544
0545
0546  def molecule_from_bytes(filename: str, data: bytes, charge_hint=0):
0547      ext = Path(filename).suffix.lower()
0548      text = None
0549
0550      if ext in {".sdf", ".sd"}:
0551          supplier = Chem.ForwardSDMolSupplier(io.BytesIO(data), removeHs=False)
0552          mol = next((m for m in supplier if m is not None), None)
0553      elif ext in {".mol", ".mdl"}:
0554          text = data.decode("utf-8", errors="replace")
0555          mol = Chem.MolFromMolBlock(text, removeHs=False, sanitize=True)
0556      elif ext == ".mol2":
0557          text = data.decode("utf-8", errors="replace")
0558          mol = Chem.MolFromMol2Block(text, removeHs=False, sanitize=True)
0559      elif ext in {".pdb", ".ent"}:
0560          text = data.decode("utf-8", errors="replace")
0561          mol = Chem.MolFromPDBBlock(text, removeHs=False, sanitize=True)
0562      elif ext in {".xyz"}:
0563          text = data.decode("utf-8", errors="replace")
0564          mol = Chem.MolFromXYZBlock(text)
0565          mol = infer_bonds_xyz(mol, charge=charge_hint)
0566      elif ext in {".smi", ".smiles", ".can"}:
0567          text = data.decode("utf-8", errors="replace").strip()
0568          token = text.splitlines()[0].strip().split()[0]
0569          mol = Chem.MolFromSmiles(token)
0570      elif ext in {".inchi"}:
0571          text = data.decode("utf-8", errors="replace").strip().splitlines()[0]
0572          mol = Chem.MolFromInchi(text)
0573      elif ext in {".txt"}:
0574          text = data.decode("utf-8", errors="replace").strip()
0575          first = text.splitlines()[0].strip()
0576          mol = Chem.MolFromInchi(first) if first.startswith("InChI=") else
Chem.MolFromSmiles(first.split()[0])
0577      else:
0578          mol = mol_from_openbabel_bytes(filename, data)
0579
0580      if mol is None:
0581          # Fallback terakhir untuk format yang RDKit gagal baca.
0582          mol = mol_from_openbabel_bytes(filename, data)
0583      return mol
0584
0585
0586  def molecule_from_local_path(path_text: str, charge_hint=0):
0587      path = Path(path_text).expanduser()
0588      if not path.exists() or not path.is_file():
0589          raise FileNotFoundError(f"File tidak ditemukan: {path}")
0590      return molecule_from_bytes(path.name, path.read_bytes(),
charge_hint=charge_hint)
0591
0592
0593  def molecule_from_identifier(identifier: str):
0594      s = identifier.strip()
0595      if not s:
0596          raise ValueError("SMILES/InChI kosong.")
0597      if s.startswith("InChI="):
0598          mol = Chem.MolFromInchi(s)
0599      else:
0600          mol = Chem.MolFromSmiles(s)
0601      if mol is None:
0602          raise ValueError(f"SMILES/InChI tidak valid: {s}")
```

```
0603      return mol
0604
0605
0606  def molecule_identifiers(mol):
0607      heavy = Chem.RemoveHs(Chem.Mol(mol))
0608      formula = rdMolDescriptors.CalcMolFormula(heavy)
0609      try:
0610          canonical = Chem.MolToSmiles(heavy, canonical=True, isomericSmiles=True)
0611      except Exception:
0612          canonical = ""
0613      try:
0614          inchi = Chem.MolToInchi(heavy)
0615      except Exception:
0616          inchi = ""
0617      try:
0618          inchikey = Chem.MolToInchiKey(heavy)
0619      except Exception:
0620          inchikey = ""
0621      try:
0622          mw = float(Descriptors.MolWt(heavy))
0623          exact = float(rdMolDescriptors.CalcExactMolWt(heavy))
0624      except Exception:
0625          mw = float("nan")
0626          exact = float("nan")
0627      return {
0628          "formula": formula,
0629          "smiles": canonical,
0630          "inchi": inchi,
0631          "inchikey": inchikey,
0632          "mw": mw,
0633          "exact_mass": exact,
0634      }
0635
0636
0637  def formal_charge(mol):
0638      try:
0639          return int(sum(a.GetFormalCharge() for a in mol.GetAtoms()))
0640      except Exception:
0641          return 0
0642
0643
0644  def mol_to_pyscf_atoms(rdkit_mol):
0645      conf = rdkit_mol.GetConformer()
0646      atoms = []
0647      for i, atom in enumerate(rdkit_mol.GetAtoms()):
0648          p = conf.GetAtomPosition(i)
0649          atoms.append([atom.GetSymbol(), (float(p.x), float(p.y), float(p.z))])
0650      return atoms
0651
0652
0653  def rdkit_with_new_coords(template_mol, coords_angstrom):
0654      mol = Chem.Mol(template_mol)
0655      if mol.GetNumConformers() == 0:
0656          conf = Chem.Conformer(mol.GetNumAtoms())
0657          mol.AddConformer(conf, assignId=True)
0658      conf = mol.GetConformer()
0659      conf.Set3D(True)
0660      coords = np.asarray(coords_angstrom, dtype=float)
0661      if len(coords) != mol.GetNumAtoms():
0662          raise ValueError("Jumlah koordinat final tidak sama dengan jumlah atom pada
struktur awal.")
0663      for i, (x, y, z) in enumerate(coords):
0664          conf.SetAtomPosition(i, (float(x), float(y), float(z)))
```

```
    return mol


def mol_to_xyz(mol, title="molecule"):
    conf = mol.GetConformer()
    lines = [str(mol.GetNumAtoms()), title]
    for i, atom in enumerate(mol.GetAtoms()):
        p = conf.GetAtomPosition(i)
        lines.append(f"{atom.GetSymbol():2s} {p.x: .10f} {p.y: .10f} {p.z: .10f}")
    return "\n".join(lines) + "\n"


def export_bonded_formats(mol, title="optimized molecule"):
    mol = Chem.Mol(mol)
    if mol.HasProp("_Name"):
        mol.ClearProp("_Name")
    mol.SetProp("_Name", title)
    molblock = Chem.MolToMolBlock(mol)
    sdf = molblock + "\n$$$$\n"
    try:
        pdb = Chem.MolToPDBBlock(mol)
    except Exception:
        pdb = ""
    xyz = mol_to_xyz(mol, title=title)
    return {"mol": molblock + "\n", "sdf": sdf, "pdb": pdb, "xyz": xyz}




def _as_json_value(value):
    """Konversi scalar/array NumPy menjadi objek JSON yang aman."""
    if value is None:
        return None
    if isinstance(value, (str, bool, int, float)):
        return value
    if isinstance(value, np.generic):
        return value.item()
    if isinstance(value, np.ndarray):
        return value.tolist()
    if isinstance(value, (tuple, list)):
        return [_as_json_value(v) for v in value]
    try:
        return float(value)
    except Exception:
        return str(value)


def calculate_geometry_tables(rdmol):
    """Semua panjang ikatan, sudut ikatan, dan dihedral dari geometri final."""
    from rdkit.Chem import rdMolTransforms

    mol = Chem.Mol(rdmol)
    conf = mol.GetConformer()
    bonds = []
    for bond in mol.GetBonds():
        i = bond.GetBeginAtomIdx()
        j = bond.GetEndAtomIdx()
        try:
            length = float(rdMolTransforms.GetBondLength(conf, i, j))
        except Exception:
            pi = conf.GetAtomPosition(i)
            pj = conf.GetAtomPosition(j)
            length = float(math.sqrt((pi.x-pj.x)**2 + (pi.y-pj.y)**2 + (pi.z-
```

```
pj.z)**2))
0728          bonds.append({
0729              "Bond": len(bonds) + 1,
0730              "Atom 1": i + 1,
0731              "Unsur 1": mol.GetAtomWithIdx(i).GetSymbol(),
0732              "Atom 2": j + 1,
0733              "Unsur 2": mol.GetAtomWithIdx(j).GetSymbol(),
0734              "Panjang (Å)": length,
0735              "Orde ikatan": float(bond.GetBondTypeAsDouble()),
0736              "Aromatik": bool(bond.GetIsAromatic()),
0737          })
0738
0739      angles = []
0740      for j in range(mol.GetNumAtoms()):
0741          neigh = sorted(a.GetIdx() for a in mol.GetAtomWithIdx(j).GetNeighbors())
0742          for a in range(len(neigh)):
0743              for b in range(a + 1, len(neigh)):
0744                  i, k = neigh[a], neigh[b]
0745                  try:
0746                      angle = float(rdMolTransforms.GetAngleDeg(conf, i, j, k))
0747                  except Exception:
0748                      continue
0749                  angles.append({
0750                      "Angle": len(angles) + 1,
0751                      "Atom 1": i + 1,
0752                      "Unsur 1": mol.GetAtomWithIdx(i).GetSymbol(),
0753                      "Atom 2 (pusat)": j + 1,
0754                      "Unsur 2": mol.GetAtomWithIdx(j).GetSymbol(),
0755                      "Atom 3": k + 1,
0756                      "Unsur 3": mol.GetAtomWithIdx(k).GetSymbol(),
0757                      "Sudut (deg)": angle,
0758                  })
0759
0760      dihedrals = []
0761      seen = set()
0762      for bond in mol.GetBonds():
0763          j = bond.GetBeginAtomIdx()
0764          k = bond.GetEndAtomIdx()
0765          n_j = [a.GetIdx() for a in mol.GetAtomWithIdx(j).GetNeighbors() if
a.GetIdx() != k]
0766          n_k = [a.GetIdx() for a in mol.GetAtomWithIdx(k).GetNeighbors() if
a.GetIdx() != j]
0767          for i in n_j:
0768              for l in n_k:
0769                  if len({i, j, k, l}) < 4:
0770                      continue
0771                  key1 = (i, j, k, l)
0772                  key2 = (l, k, j, i)
0773                  key = min(key1, key2)
0774                  if key in seen:
0775                      continue
0776                  seen.add(key)
0777                  try:
0778                      dih = float(rdMolTransforms.GetDihedralDeg(conf, i, j, k, l))
0779                  except Exception:
0780                      continue
0781                  dihedrals.append({
0782                      "Dihedral": len(dihedrals) + 1,
0783                      "Atom 1": i + 1,
0784                      "Unsur 1": mol.GetAtomWithIdx(i).GetSymbol(),
0785                      "Atom 2": j + 1,
0786                      "Unsur 2": mol.GetAtomWithIdx(j).GetSymbol(),
0787                      "Atom 3": k + 1,
```

```
                    "Unsur 3": mol.GetAtomWithIdx(k).GetSymbol(),
                    "Atom 4": l + 1,
                    "Unsur 4": mol.GetAtomWithIdx(l).GetSymbol(),
                    "Dihedral (deg)": dih,
                })
    return bonds, angles, dihedrals


def calculate_rdkit_qsar_qspr_descriptors(rdmol):
    """Koleksi deskriptor RDKit 2D + 3D untuk QSAR/QSPR secara luas."""
    mol3d = Chem.Mol(rdmol)
    try:
        Chem.SanitizeMol(mol3d)
    except Exception:
        pass
    try:
        mol2d = Chem.RemoveHs(Chem.Mol(mol3d))
    except Exception:
        mol2d = Chem.Mol(mol3d)

    values = {}
    # Seluruh scalar descriptor bawaan RDKit (umumnya >200 descriptor pada rilis modern).
    for name, fn in getattr(Descriptors, "descList", []):
        try:
            v = fn(mol2d)
            if np.isscalar(v):
                fv = float(v)
                if math.isfinite(fv):
                    values[name] = (fv, "RDKit 2D")
        except Exception:
            pass

    # Deskriptor 3D penting untuk QSPR/QSAR konformasi.
    three_d = {
        "Asphericity": getattr(rdMolDescriptors, "CalcAsphericity", None),
        "Eccentricity": getattr(rdMolDescriptors, "CalcEccentricity", None),
        "InertialShapeFactor": getattr(rdMolDescriptors, "CalcInertialShapeFactor", None),
        "NPR1": getattr(rdMolDescriptors, "CalcNPR1", None),
        "NPR2": getattr(rdMolDescriptors, "CalcNPR2", None),
        "PMI1": getattr(rdMolDescriptors, "CalcPMI1", None),
        "PMI2": getattr(rdMolDescriptors, "CalcPMI2", None),
        "PMI3": getattr(rdMolDescriptors, "CalcPMI3", None),
        "RadiusOfGyration": getattr(rdMolDescriptors, "CalcRadiusOfGyration", None),
        "SpherocityIndex": getattr(rdMolDescriptors, "CalcSpherocityIndex", None),
    }
    if mol3d.GetNumConformers():
        for name, fn in three_d.items():
            if fn is None:
                continue
            try:
                values[name] = (float(fn(mol3d)), "RDKit 3D")
            except Exception:
                pass
        try:
            values["MolecularVolume_A3"] = (float(AllChem.ComputeMolVolume(mol3d)), "RDKit 3D")
        except Exception:
            pass

    # Gasteiger charge summary berguna sebagai descriptor cepat non-QM.
    try:
```

```
0848          from rdkit.Chem import rdPartialCharges
0849          gm = Chem.Mol(mol3d)
0850          rdPartialCharges.ComputeGasteigerCharges(gm)
0851          q = np.asarray([float(a.GetProp("_GasteigerCharge")) for a in
gm.GetAtoms()], dtype=float)
0852          q = q[np.isfinite(q)]
0853          if q.size:
0854              values["GasteigerCharge_Min"] = (float(np.min(q)), "Partial charge")
0855              values["GasteigerCharge_Max"] = (float(np.max(q)), "Partial charge")
0856              values["GasteigerCharge_Mean"] = (float(np.mean(q)), "Partial charge")
0857              values["GasteigerCharge_AbsSum"] = (float(np.sum(np.abs(q))), "Partial
charge")
0858      except Exception:
0859          pass
0860
0861      rows = [
0862          {"Descriptor": name, "Value": val, "Category": category}
0863          for name, (val, category) in sorted(values.items(), key=lambda kv:
kv[0].lower())
0864      ]
0865      return rows
0866
0867
0868  def add_qm_qsar_descriptors(rows, result):
0869      """Tambahkan descriptor konseptual-DFT/QM ke tabel QSAR/QSPR."""
0870      out = list(rows or [])
0871      def add(name, value, category="QM / conceptual DFT"):
0872          if value is None:
0873              return
0874          try:
0875              fv = float(value)
0876              if math.isfinite(fv):
0877                  out.append({"Descriptor": name, "Value": fv, "Category": category})
0878          except Exception:
0879              pass
0880
0881      add("QM_TotalEnergy_Hartree", result.get("energy_final_h"), "QM")
0882      add("QM_HOMO_eV", result.get("homo_ev"), "QM")
0883      add("QM_LUMO_eV", result.get("lumo_ev"), "QM")
0884      add("QM_HOMO_LUMO_Gap_eV", result.get("gap_ev"), "QM")
0885      add("QM_Dipole_X_D", result.get("dipole_x_d"), "QM")
0886      add("QM_Dipole_Y_D", result.get("dipole_y_d"), "QM")
0887      add("QM_Dipole_Z_D", result.get("dipole_z_d"), "QM")
0888      add("QM_Dipole_Magnitude_D", result.get("dipole_total_d"), "QM")
0889
0890      h = result.get("homo_ev")
0891      l = result.get("lumo_ev")
0892      if h is not None and l is not None:
0893          try:
0894              h = float(h); l = float(l)
0895              ip = -h
0896              ea = -l
0897              chi = (ip + ea) / 2.0
0898              eta = (ip - ea) / 2.0
0899              mu = -chi
0900              softness = (1.0 / (2.0 * eta)) if abs(eta) > 1e-12 else None
0901              omega = (mu * mu / (2.0 * eta)) if abs(eta) > 1e-12 else None
0902              add("Koopmans_IonizationPotential_eV", ip)
0903              add("Koopmans_ElectronAffinity_eV", ea)
0904              add("Electronegativity_eV", chi)
0905              add("ChemicalPotential_eV", mu)
0906              add("GlobalHardness_eV", eta)
0907              add("GlobalSoftness_1_per_eV", softness)
```

```
0908              add("ElectrophilicityIndex_eV", omega)
0909          except Exception:
0910              pass
0911
0912      mep = result.get("mep_stats") or {}
0913      add("MEP_Min_au", mep.get("min"), "QM grid")
0914      add("MEP_Max_au", mep.get("max"), "QM grid")
0915      add("MEP_Mean_au", mep.get("mean"), "QM grid")
0916      return out
0917
0918
0919  def build_optimization_geometry_energy_rows(result):
0920      rows = []
0921      for r in result.get("bond_lengths", []):
0922          rows.append({
0923              "Jenis": "Bond", "Parameter": f"{r.get('Atom 1')}-{r.get('Atom 2')}",
0924              "Nilai": r.get("Panjang (Å)"), "Satuan": "Å",
0925              "Detail": f"{r.get('Unsur 1')}-{r.get('Unsur 2')}; order={r.get('Orde
ikatan')}; aromatic={r.get('Aromatik')}",
0926          })
0927      for r in result.get("bond_angles", []):
0928          rows.append({
0929              "Jenis": "Angle", "Parameter": f"{r.get('Atom 1')}-{r.get('Atom 2
(pusat)')}-{r.get('Atom 3')}",
0930              "Nilai": r.get("Sudut (deg)"), "Satuan": "deg",
0931              "Detail": f"{r.get('Unsur 1')}-{r.get('Unsur 2')}-{r.get('Unsur 3')}",
0932          })
0933      for r in result.get("dihedral_angles", []):
0934          rows.append({
0935              "Jenis": "Dihedral", "Parameter": f"{r.get('Atom 1')}-{r.get('Atom 2')}-
{r.get('Atom 3')}-{r.get('Atom 4')}",
0936              "Nilai": r.get("Dihedral (deg)"), "Satuan": "deg",
0937              "Detail": f"{r.get('Unsur 1')}-{r.get('Unsur 2')}-{r.get('Unsur 3')}-
{r.get('Unsur 4')}",
0938          })
0939      e = result.get("energy") or {}
0940      for key, unit in (("hartree", "Eh"), ("ev", "eV"), ("kj_mol", "kJ/mol"),
("kcal_mol", "kcal/mol"), ("cm-1", "cm-1")):
0941          if e.get(key) is not None:
0942              rows.append({"Jenis": "Energy", "Parameter": f"Total energy ({key})",
"Nilai": e.get(key), "Satuan": unit, "Detail": "Final electronic energy"})
0943      de = result.get("delta_energy") or {}
0944      for key, unit in (("hartree", "Eh"), ("ev", "eV"), ("kj_mol", "kJ/mol"),
("kcal_mol", "kcal/mol")):
0945          if de.get(key) is not None:
0946              rows.append({"Jenis": "Energy", "Parameter": f"Optimization ΔE ({key})",
"Nilai": de.get(key), "Satuan": unit, "Detail": "E_final - E_initial"})
0947      for item in result.get("optimization_energy_history", []):
0948          step = item.get("Step")
0949          for k, unit in (("Energy (Eh)", "Eh"), ("Energy (eV)", "eV"), ("Energy
(kJ/mol)", "kJ/mol"), ("Energy (kcal/mol)", "kcal/mol")):
0950              if item.get(k) is not None:
0951                  rows.append({"Jenis": "Optimization step energy", "Parameter":
f"Step {step}", "Nilai": item.get(k), "Satuan": unit, "Detail": "Energy recorded by
geometry optimizer callback"})
0952      for orb in result.get("orbitals", []):
0953          if orb.get("Energi (eV)") is not None:
0954              rows.append({
0955                  "Jenis": "Orbital energy",
0956                  "Parameter": f"MO {orb.get('Spin','')} {orb.get('MO','')}",
0957                  "Nilai": orb.get("Energi (eV)"), "Satuan": "eV",
0958                  "Detail": f"Occupancy={orb.get('Occupancy')}"
0959              })
```

```
0960      return rows
0961
0962
0963  def enrich_result_common(result):
0964      """Pengayaan non-destruktif untuk semua backend: geometri + QSAR/QSPR."""
0965      result.setdefault("addon_meta", {"name": result.get("addon", "None"), "status":
"backend_not_implemented" if result.get("addon", "None") != "None" else "not_requested"})
0966      result.setdefault("addon_results", [])
0967      result.setdefault("thermochemistry", [])
0968      result.setdefault("cube_files", {})
0969      result.setdefault("mep_stats", {})
0970      try:
0971          mol = Chem.MolFromMolBlock(result.get("mol", ""), removeHs=False,
sanitize=False)
0972          if mol is None:
0973              return result
0974          bonds, angles, dihedrals = calculate_geometry_tables(mol)
0975          result["bond_lengths"] = bonds
0976          result["bond_angles"] = angles
0977          result["dihedral_angles"] = dihedrals
0978          result["qsar_qspr_descriptors"] = add_qm_qsar_descriptors(
0979              calculate_rdkit_qsar_qspr_descriptors(mol), result
0980          )
0981          result["optimization_geometry_energies"] =
build_optimization_geometry_energy_rows(result)
0982      except Exception as exc:
0983          result.setdefault("analysis_warnings", []).append(f"Analisis geometri/QSAR
gagal: {exc}")
0984      return result
0985
0986
0987  def _spin_summed_dm(mf):
0988      dm = mf.make_rdm1()
0989      if isinstance(dm, (tuple, list)):
0990          return np.asarray(dm[0]) + np.asarray(dm[1])
0991      arr = np.asarray(dm)
0992      if arr.ndim == 3 and arr.shape[0] == 2:
0993          return arr[0] + arr[1]
0994      return arr
0995
0996
0997  def _frontier_mo_coefficients(mf):
0998      e = mf.mo_energy
0999      occ = mf.mo_occ
1000      coeff = mf.mo_coeff
1001      channels = []
1002      if isinstance(e, (tuple, list)) or (isinstance(e, np.ndarray) and
np.asarray(e).ndim == 2):
1003          labels = ["alpha", "beta"]
1004          for label, en, oc, co in zip(labels, list(e), list(occ), list(coeff)):
1005              channels.append((label, np.asarray(en, dtype=float), np.asarray(oc,
dtype=float), np.asarray(co)))
1006      else:
1007          channels.append(("R", np.asarray(e, dtype=float), np.asarray(occ,
dtype=float), np.asarray(coeff)))
1008      occ_cand = []
1009      vir_cand = []
1010      for label, en, oc, co in channels:
1011          for idx in np.where(oc > 1e-10)[0]:
1012              occ_cand.append((float(en[idx]), label, int(idx), co[:, idx]))
1013          for idx in np.where(oc <= 1e-10)[0]:
1014              vir_cand.append((float(en[idx]), label, int(idx), co[:, idx]))
1015      homo = max(occ_cand, key=lambda x: x[0]) if occ_cand else None
```

```
    lumo = min(vir_cand, key=lambda x: x[0]) if vir_cand else None
    return homo, lumo


def generate_pyscf_cube_files(mf, mol, mol_dir, base, grid_points=60, progress_log=None):
    """MEP, density, HOMO dan LUMO Gaussian cube dari wavefunction final PySCF."""
    from pyscf.tools import cubegen
    mol_dir = Path(mol_dir)
    n = int(grid_points)
    dm = _spin_summed_dm(mf)
    files = {}
    stats = {}
    density_path = mol_dir / f"{base}_density.cube"
    mep_path = mol_dir / f"{base}_mep.cube"
    homo_path = mol_dir / f"{base}_homo.cube"
    lumo_path = mol_dir / f"{base}_lumo.cube"

    if progress_log:
        append_log(Path(progress_log), f"Membuat cube grid {n}x{n}x{n}: density/MEP/HOMO/LUMO.")
    try:
        dens = np.asarray(cubegen.density(mol, str(density_path), dm, nx=n, ny=n, nz=n), dtype=float)
        files["Electron density cube"] = str(density_path)
        stats["density_min"] = float(np.nanmin(dens)); stats["density_max"] = float(np.nanmax(dens))
    except Exception as exc:
        if progress_log: append_log(Path(progress_log), f"Density cube gagal: {exc}")
    try:
        mep = np.asarray(cubegen.mep(mol, str(mep_path), dm, nx=n, ny=n, nz=n), dtype=float)
        files["MEP cube"] = str(mep_path)
        finite = mep[np.isfinite(mep)]
        if finite.size:
            stats["min"] = float(np.min(finite)); stats["max"] = float(np.max(finite)); stats["mean"] = float(np.mean(finite))
    except Exception as exc:
        if progress_log: append_log(Path(progress_log), f"MEP cube gagal: {exc}")

    homo, lumo = _frontier_mo_coefficients(mf)
    if homo is not None:
        try:
            cubegen.orbital(mol, str(homo_path), np.asarray(homo[3]), nx=n, ny=n, nz=n)
            files["HOMO cube"] = str(homo_path)
            stats["homo_channel"] = homo[1]; stats["homo_mo"] = homo[2] + 1
        except Exception as exc:
            if progress_log: append_log(Path(progress_log), f"HOMO cube gagal: {exc}")
    if lumo is not None:
        try:
            cubegen.orbital(mol, str(lumo_path), np.asarray(lumo[3]), nx=n, ny=n, nz=n)
            files["LUMO cube"] = str(lumo_path)
            stats["lumo_channel"] = lumo[1]; stats["lumo_mo"] = lumo[2] + 1
        except Exception as exc:
            if progress_log: append_log(Path(progress_log), f"LUMO cube gagal: {exc}")
    return files, stats


```

```
1068  def run_pyscf_uvvis(mf, nroots=20, progress_log=None):
1069      """TDDFT/TDHF/TDA excited states + oscillator strengths untuk UV-Vis."""
1070      if progress_log: append_log(Path(progress_log), f"Add-On UV-Vis: menghitung
{int(nroots)} excited states.")
1071      td = None
1072      method_used = ""
1073      # DFT objects biasanya memiliki TDDFT; HF memiliki TDHF. TDA menjadi fallback
robust.
1074      for attr, label in (("TDDFT", "TDDFT"), ("TDHF", "TDHF"), ("TDA", "TDA")):
1075          try:
1076              factory = getattr(mf, attr)
1077              td = factory()
1078              method_used = label
1079              break
1080          except Exception:
1081              continue
1082      if td is None:
1083          raise RuntimeError("Objek mean-field ini tidak menyediakan TDDFT/TDHF/TDA.")
1084      td.nstates = int(nroots)
1085      td.kernel()
1086      energies_h = np.real(np.asarray(td.e, dtype=complex)).astype(float)
1087      try:
1088          osc = np.real(np.asarray(td.oscillator_strength(),
dtype=complex)).astype(float)
1089      except Exception:
1090          osc = np.full_like(energies_h, np.nan, dtype=float)
1091      rows = []
1092      for i, eh in enumerate(energies_h, start=1):
1093          ev = float(eh * HARTREE_TO_EV)
1094          rows.append({
1095              "State": i,
1096              "Method": method_used,
1097              "Excitation Energy (Eh)": float(eh),
1098              "Excitation Energy (eV)": ev,
1099              "Wavelength (nm)": (EV_NM / ev if ev > 1e-12 else None),
1100              "Oscillator Strength": (float(osc[i-1]) if i-1 < len(osc) and
np.isfinite(osc[i-1]) else None),
1101          })
1102      return rows
1103
1104
1105  def run_pyscf_thermochemistry(mf, mol, hessian, temperature=DEFAULT_TEMPERATURE_K,
pressure=DEFAULT_PRESSURE_PA, progress_log=None):
1106      from pyscf.hessian import thermo as pyscf_thermo
1107      if progress_log: append_log(Path(progress_log), f"Add-On Thermodynamics:
T={temperature} K, P={pressure} Pa.")
1108      harmonic = pyscf_thermo.harmonic_analysis(mol, np.asarray(hessian),
imaginary_freq=False)
1109      freq_au = np.asarray(harmonic["freq_au"])
1110      # Thermochemistry ideal-gas/harmonic hanya memakai mode vibrasi nyata positif.
1111      # Mode imajiner/zero mode tetap tersedia di tabel frekuensi utama, tetapi tidak
1112      # dimasukkan ke fungsi partisi agar log/eksponensial tidak menghasilkan NaN
palsu.
1113      freq_real = np.real(freq_au[np.isreal(freq_au)]) if np.iscomplexobj(freq_au)
else np.asarray(freq_au, dtype=float)
1114      freq_positive = np.asarray(freq_real, dtype=float)
1115      freq_positive = freq_positive[np.isfinite(freq_positive) & (freq_positive >
1.0e-12)]
1116      if freq_positive.size == 0:
1117          raise RuntimeError("Tidak ada frekuensi vibrasi positif yang dapat dipakai
untuk thermochemistry.")
1118      tdata = pyscf_thermo.thermo(mf, freq_positive, float(temperature),
float(pressure))
```

```
    rows = []
    for key, value in tdata.items():
        if isinstance(value, tuple) and len(value) == 2:
            val, unit = value
        else:
            val, unit = value, ""
        rows.append({"Property": key, "Value": _as_json_value(val), "Unit": unit})
    return rows


def run_pyscf_nmr(mf, progress_log=None):
    """NMR shielding melalui PySCF properties extension bila tersedia/kompatibel."""
    if progress_log: append_log(Path(progress_log), "Add-On NMR: memuat pyscf
properties extension.")
    try:
        ensure_python_package("pyscf.prop.nmr", "pyscf[properties]")
        from pyscf.prop import nmr
    except Exception as exc:
        raise RuntimeError(f"PySCF properties/NMR tidak tersedia: {exc}") from exc

    cls_name = mf.__class__.__name__.upper()
    candidates = []
    if "UKS" in cls_name: candidates = ["UKS", "UHF"]
    elif "RKS" in cls_name: candidates = ["RKS", "RHF"]
    elif "UHF" in cls_name: candidates = ["UHF"]
    else: candidates = ["RHF"]
    obj = None
    for name in candidates:
        cls = getattr(nmr, name, None)
        if cls is not None:
            obj = cls(mf)
            break
    if obj is None:
        raise RuntimeError(f"Driver NMR tidak ditemukan untuk {cls_name}.")
    shielding = np.asarray(obj.kernel(), dtype=float)
    rows = []
    for ia in range(shielding.shape[0]):
        tensor = shielding[ia]
        eig = np.linalg.eigvalsh(0.5 * (tensor + tensor.T))
        iso = float(np.trace(tensor) / 3.0)
        rows.append({
            "Atom": ia + 1,
            "Unsur": mf.mol.atom_symbol(ia),
            "Isotropic Shielding (ppm)": iso,
            "Eigenvalue 1 (ppm)": float(eig[0]),
            "Eigenvalue 2 (ppm)": float(eig[1]),
            "Eigenvalue 3 (ppm)": float(eig[2]),
            "Tensor xx": float(tensor[0,0]), "Tensor xy": float(tensor[0,1]),
"Tensor xz": float(tensor[0,2]),
            "Tensor yx": float(tensor[1,0]), "Tensor yy": float(tensor[1,1]),
"Tensor yz": float(tensor[1,2]),
            "Tensor zx": float(tensor[2,0]), "Tensor zy": float(tensor[2,1]),
"Tensor zz": float(tensor[2,2]),
        })
    return rows


def _write_xyz_body(rdmol):
    conf = rdmol.GetConformer()
    lines = []
    for i, atom in enumerate(rdmol.GetAtoms()):
        p = conf.GetAtomPosition(i)
        lines.append(f"{atom.GetSymbol():2s} {p.x: .10f} {p.y: .10f} {p.z: .10f}")
```

```
1178      return lines
1179
1180
1181  def run_orca_raman_addon(rdmol, config, spec, mol_dir, stop_event, progress_log):
1182      """Raman ORCA 6.1+: Freq + analytical polarizability derivative untuk HF/DFT."""
1183      mol_dir = Path(mol_dir)
1184      inp = mol_dir / "addon_raman_orca.inp"
1185      method = config.get("method", "B3LYP")
1186      # Raman adapter memakai ORCA. Bila nama metode PySCF tidak tersedia pada
1187      # daftar ORCA aplikasi, gunakan B3LYP alih-alih menghasilkan input invalid.
1188      if method not in ORCA_METHODS:
1189          method = "B3LYP"
1190      basis = config.get("basis", "def2-SVP")
1191      charge = int(spec["charge"]); mult = int(spec["multiplicity"])
1192      nprocs = int(config.get("threads", 1)); maxcore = max(100,
int(config.get("max_memory_mb", 4000) // max(1, nprocs)))
1193      lines = [
1194          f"! {method} {basis} Freq TightSCF",
1195          f"%pal nprocs {nprocs} end",
1196          f"%maxcore {maxcore}",
1197          "%elprop Polar 1 end",
1198          f"* xyz {charge} {mult}",
1199          *_write_xyz_body(rdmol),
1200          "*",
1201      ]
1202      inp.write_text("\n".join(lines) + "\n", encoding="utf-8")
1203      orca = shutil.which("orca")
1204      if not orca:
1205          return {"status": "input_generated", "message": "ORCA tidak ditemukan; input
Raman sudah dibuat.", "input_file": str(inp)}, []
1206      outlog = mol_dir / "addon_raman_orca.out"
1207      run_process_interruptible([orca, str(inp)], mol_dir, outlog, stop_event)
1208      text = outlog.read_text(encoding="utf-8", errors="replace") if outlog.exists()
else ""
1209      rows = []
1210      # ORCA: mode: freq Activity Depolarization
1211      for m in re.finditer(r"^\s*(\d+)\s*:\s*(-?\d+(?:\.\d+)?)\s+(-
?\d+(?:\.\d+)?)\s+(-?\d+(?:\.\d+)?)\s*$", text, flags=re.M):
1212          try:
1213              rows.append({"Mode": int(m.group(1)), "Frequency (cm-1)":
float(m.group(2)), "Raman Activity": float(m.group(3)), "Depolarization":
float(m.group(4))})
1214          except Exception:
1215              pass
1216      return {"status": "completed", "message": f"ORCA Raman selesai; {len(rows)}
baris diparsing.", "input_file": str(inp), "output_file": str(outlog)}, rows
1217
1218
1219  def run_g3mp2_addon(rdmol, spec, mol_dir, stop_event, progress_log):
1220      """Hook G3MP2: jalankan Gaussian bila ada; bila tidak, simpan input tanpa
memalsukan ΔHf."""
1221      mol_dir = Path(mol_dir)
1222      gjf = mol_dir / "addon_g3mp2.gjf"
1223      charge = int(spec["charge"]); mult = int(spec["multiplicity"])
1224      lines = [
1225          "%chk=addon_g3mp2.chk",
1226          "#p G3MP2",
1227          "",
1228          "G3MP2 add-on generated by QM KOMPUTASI V2.3.4",
1229          "",
1230          f"{charge} {mult}",
1231          *_write_xyz_body(rdmol),
1232          "",
```

```
1233      ]
1234      gjf.write_text("\n".join(lines) + "\n", encoding="utf-8")
1235      exe = next((shutil.which(x) for x in ("g16", "g09", "gaussian") if
shutil.which(x)), None)
1236      if not exe:
1237          return {"status": "input_generated", "message": "Gaussian G3MP2 executable
tidak ditemukan; input .gjf sudah dibuat. ΔHf tidak direkayasa.", "input_file": str(gjf)}
1238      outlog = mol_dir / "addon_g3mp2.out"
1239      # Gaussian umum menerima input melalui stdin.
1240      with open(gjf, "rb") as fin, open(outlog, "wb") as fout:
1241          proc = subprocess.Popen([exe], cwd=str(mol_dir), stdin=fin, stdout=fout,
stderr=subprocess.STDOUT)
1242          while proc.poll() is None:
1243              if stop_event.is_set():
1244                  proc.terminate()
1245                  raise StopRequested("G3MP2 dihentikan pengguna.")
1246              time.sleep(0.25)
1247      text = outlog.read_text(encoding="utf-8", errors="replace")
1248      result = {"status": "completed" if proc.returncode == 0 else "failed",
"input_file": str(gjf), "output_file": str(outlog)}
1249      patterns = {
1250          "g3mp2_energy_h": r"G3MP2(?:\s+Energy)?\s*=\s*(-?\d+\.\d+(?:[DEde][+-
]?\d+)?)",
1251          "g3mp2_enthalpy_h": r"G3MP2\s+Enthalpy\s*=\s*(-?\d+\.\d+(?:[DEde][+-
]?\d+)?)",
1252          "delta_hf": r"(?:Delta\s*Hf|Heat\s+of\s+Formation)\s*=\s*(-
?\d+\.\d+(?:[DEde][+-]?\d+)?)",
1253      }
1254      for key, pat in patterns.items():
1255          mm = re.findall(pat, text, flags=re.I)
1256          if mm:
1257              try: result[key] = float(mm[-1].replace("D", "E").replace("d", "e"))
1258              except Exception: pass
1259      result["message"] = "G3MP2 selesai. Nilai ΔHf hanya diisi bila output Gaussian
menyatakannya eksplisit."
1260      return result
1261
1262
1263  def run_nbo_addon(rdmol, spec, config, mol_dir, stop_event, progress_log):
1264      """NBO def2-TZPP melalui ORCA + standalone NBO6/7 jika NBOEXE tersedia."""
1265      mol_dir = Path(mol_dir)
1266      inp = mol_dir / "addon_nbo_def2-TZPP.inp"
1267      method = config.get("method", "B3LYP")
1268      # Jika metode utama bukan HF/DFT yang lazim di ORCA, gunakan B3LYP sebagai
adapter NBO yang eksplisit.
1269      if method not in ORCA_METHODS:
1270          method = "B3LYP"
1271      charge = int(spec["charge"]); mult = int(spec["multiplicity"])
1272      nprocs = int(config.get("threads", 1)); maxcore = max(100,
int(config.get("max_memory_mb", 4000) // max(1, nprocs)))
1273      lines = [
1274          f"! {method} def2-TZPP SP TightSCF NBO",
1275          f"%pal nprocs {nprocs} end",
1276          f"%maxcore {maxcore}",
1277          f"* xyz {charge} {mult}",
1278          *_write_xyz_body(rdmol),
1279          "*",
1280      ]
1281      inp.write_text("\n".join(lines) + "\n", encoding="utf-8")
1282      orca = shutil.which("orca")
1283      nboexe = os.environ.get("NBOEXE", "").strip()
1284      if not orca or not nboexe or not Path(nboexe).exists():
1285          missing = []
```

```
1286          if not orca: missing.append("ORCA")
1287          if not nboexe or not Path(nboexe).exists(): missing.append("NBOEXE/NBO6-7")
1288          return {"status": "input_generated", "message": f"Input NBO dibuat; belum
dijalankan karena {', '.join(missing)} tidak tersedia.", "input_file": str(inp), "basis":
"def2-TZPP"}
1289      outlog = mol_dir / "addon_nbo_def2-TZPP.out"
1290      run_process_interruptible([orca, str(inp)], mol_dir, outlog, stop_event)
1291      return {"status": "completed", "message": "ORCA/NBO selesai; lihat output
lengkap untuk NPA/NBO/BD* dan donor-acceptor.", "input_file": str(inp), "output_file":
str(outlog), "basis": "def2-TZPP"}
1292
1293
1294  def apply_selected_addon_pyscf(addon, mf, mol, rdmol, spec, config, mol_dir,
stop_event, progress_log, hessian=None):
1295      addon = addon or "None"
1296      meta = {"name": addon, "status": "not_requested" if addon == "None" else
"started"}
1297      rows = []
1298      thermo_rows = []
1299      if addon == "None":
1300          return meta, rows, thermo_rows, hessian
1301      try:
1302          if addon == "IR":
1303              if hessian is None:
1304                  hessian = np.asarray(mf.Hessian().kernel(), dtype=float)
1305              rows = _pyscf_hessian_to_frequencies(mol, hessian)
1306              meta.update(
1307                  status="completed",
1308                  message=f"IR harmonik selesai; {len(rows)} mode vibrasi dihitung
tanpa thermochemistry.",
1309              )
1310          elif addon == "Thermodynamics":
1311              if hessian is None:
1312                  hessian = np.asarray(mf.Hessian().kernel(), dtype=float)
1313              thermo_rows = run_pyscf_thermochemistry(
1314                  mf, mol, hessian,
1315                  temperature=float(config.get("temperature_k",
DEFAULT_TEMPERATURE_K)),
1316                  pressure=float(config.get("pressure_pa", DEFAULT_PRESSURE_PA)),
1317                  progress_log=progress_log,
1318              )
1319              meta.update(status="completed", message="Thermodynamics harmonik
selesai.")
1320          elif addon == "UV-Vis":
1321              rows = run_pyscf_uvvis(mf, int(config.get("uvvis_nroots", 20)),
progress_log)
1322              meta.update(status="completed", message=f"{len(rows)} excited states
dihitung.")
1323          elif addon == "NMR":
1324              rows = run_pyscf_nmr(mf, progress_log)
1325              meta.update(status="completed", message=f"Shielding NMR dihitung untuk
{len(rows)} atom.")
1326          elif addon == "Raman":
1327              meta, rows = run_orca_raman_addon(rdmol, config, spec, mol_dir,
stop_event, progress_log)
1328              meta["name"] = addon
1329          elif addon == "ΔH f (g3mp2)":
1330              meta = run_g3mp2_addon(rdmol, spec, mol_dir, stop_event, progress_log)
1331              meta["name"] = addon
1332          elif addon == "NBO (def2-tzpp)":
1333              meta = run_nbo_addon(rdmol, spec, config, mol_dir, stop_event,
progress_log)
1334              meta["name"] = addon
```

```
    except Exception as exc:
        meta.update(status="failed", message=str(exc))
        append_log(Path(progress_log), f"Add-On {addon} gagal: {exc}")
    return meta, rows, thermo_rows, hessian

def mol_3d_html(mol, width=700, height=430):
    try:
        ensure_python_package("py3Dmol", "py3Dmol")
        import py3Dmol
        view = py3Dmol.view(width=width, height=height)
        view.addModel(Chem.MolToMolBlock(mol), "mol")
        view.setStyle({"stick": {}, "sphere": {"scale": 0.28}})
        view.zoomTo()
        view.setBackgroundColor("white")
        return view._make_html()
    except Exception:
        return None


# ============================================================
# FUNGSI OUTPUT PySCF (FITUR LAMA DIPERTAHANKAN + DIPERLUAS)
# ============================================================
def pyscf_xyz(mol, title: str) -> str:
    coords = mol.atom_coords(unit="Angstrom")
    lines = [str(mol.natm), title]
    for i in range(mol.natm):
        sym = mol.atom_symbol(i)
        x, y, z = coords[i]
        lines.append(f"{sym:2s} {x: .10f} {y: .10f} {z: .10f}")
    return "\n".join(lines) + "\n"


def coordinates_dataframe(mol):
    coords = mol.atom_coords(unit="Angstrom")
    rows = []
    for i in range(mol.natm):
        rows.append({
            "Atom": i + 1,
            "Unsur": mol.atom_symbol(i),
            "X (Å)": float(coords[i, 0]),
            "Y (Å)": float(coords[i, 1]),
            "Z (Å)": float(coords[i, 2]),
        })
    return pd.DataFrame(rows)


def orbital_dataframe(mf):
    e = mf.mo_energy
    occ = mf.mo_occ
    rows = []
    if isinstance(e, (tuple, list)) or (isinstance(e, np.ndarray) and
np.asarray(e).ndim == 2):
        energies = list(e)
        occs = list(occ)
        labels = ["α", "β"]
        for spin_label, en, oc in zip(labels, energies, occs):
            en = np.asarray(en, dtype=float) * HARTREE_TO_EV
            oc = np.asarray(oc, dtype=float)
            for i, (ee, oo) in enumerate(zip(en, oc), start=1):
                rows.append({"Spin": spin_label, "MO": i, "Occupancy": float(oo),
"Energi (eV)": float(ee)})
    else:
        en = np.asarray(e, dtype=float) * HARTREE_TO_EV
```

```
1396          oc = np.asarray(occ, dtype=float)
1397          for i, (ee, oo) in enumerate(zip(en, oc), start=1):
1398              rows.append({"Spin": "R", "MO": i, "Occupancy": float(oo), "Energi
(eV)": float(ee)})
1399      return pd.DataFrame(rows)
1400
1401
1402  def calculate_frontier_orbitals(mf):
1403      e = mf.mo_energy
1404      occ = mf.mo_occ
1405      candidates_occ = []
1406      candidates_vir = []
1407
1408      if isinstance(e, (tuple, list)) or (isinstance(e, np.ndarray) and
np.asarray(e).ndim == 2):
1409          for label, en, oc in zip(["α", "β"], list(e), list(occ)):
1410              en = np.asarray(en, dtype=float)
1411              oc = np.asarray(oc, dtype=float)
1412              for idx in np.where(oc > 0)[0]:
1413                  candidates_occ.append((float(en[idx]), label, int(idx + 1)))
1414              for idx in np.where(oc == 0)[0]:
1415                  candidates_vir.append((float(en[idx]), label, int(idx + 1)))
1416      else:
1417          en = np.asarray(e, dtype=float)
1418          oc = np.asarray(occ, dtype=float)
1419          for idx in np.where(oc > 0)[0]:
1420              candidates_occ.append((float(en[idx]), "R", int(idx + 1)))
1421          for idx in np.where(oc == 0)[0]:
1422              candidates_vir.append((float(en[idx]), "R", int(idx + 1)))
1423
1424      if not candidates_occ or not candidates_vir:
1425          return {
1426              "homo_index": None, "lumo_index": None,
1427              "homo_ev": None, "lumo_ev": None, "gap_ev": None,
1428          }
1429
1430      homo = max(candidates_occ, key=lambda x: x[0])
1431      lumo = min(candidates_vir, key=lambda x: x[0])
1432      return {
1433          "homo_index": f"{homo[1]}{homo[2]}",
1434          "lumo_index": f"{lumo[1]}{lumo[2]}",
1435          "homo_ev": homo[0] * HARTREE_TO_EV,
1436          "lumo_ev": lumo[0] * HARTREE_TO_EV,
1437          "gap_ev": (lumo[0] - homo[0]) * HARTREE_TO_EV,
1438      }
1439
1440
1441  def mulliken_dataframe(mf):
1442      try:
1443          _, charges = mf.mulliken_pop(verbose=0)
1444          charges = np.asarray(charges, dtype=float)
1445          rows = []
1446          for i in range(mf.mol.natm):
1447              rows.append({
1448                  "Atom": i + 1,
1449                  "Unsur": mf.mol.atom_symbol(i),
1450                  "Muatan Mulliken (e)": float(charges[i]),
1451              })
1452          return pd.DataFrame(rows)
1453      except Exception:
1454          return pd.DataFrame(columns=["Atom", "Unsur", "Muatan Mulliken (e)"])
1455
1456
```

```
1457  def make_pyscf_molecule(atoms, basis="def2-SVP", charge=0, spin=0,
max_memory_mb=4000, log_handle=None):
1458      kwargs = dict(
1459          atom=atoms,
1460          unit="Angstrom",
1461          basis=basis,
1462          charge=int(charge),
1463          spin=int(spin),
1464          verbose=4 if log_handle else 0,
1465          max_memory=int(max_memory_mb),
1466      )
1467      mol = gto.M(**kwargs)
1468      if log_handle is not None:
1469          mol.stdout = log_handle
1470      return mol
1471
1472
1473  def make_pyscf_method(mol, method_name, grid_level=3, conv_tol=1e-9, max_cycle=120,
1474                        density_fitting=False, chkfile=None, log_handle=None,
stop_event=None,
1475                        progress_log=None, use_gpu=False, gpu_threads=1):
1476      meta = PYSCF_METHODS[method_name]
1477      family = meta["family"]
1478
1479      if family == "HF" or family == "MP2":
1480          ref = meta.get("hf_reference", "AUTO")
1481          if family == "MP2":
1482              ref = "AUTO"
1483          if ref == "RHF":
1484              if mol.spin != 0:
1485                  raise ValueError("RHF memerlukan closed-shell: spin=0 /
multiplicitas 1. Gunakan ROHF atau UHF untuk open-shell.")
1486              mf = scf.RHF(mol)
1487          elif ref == "ROHF":
1488              mf = scf.ROHF(mol)
1489          elif ref == "UHF":
1490              mf = scf.UHF(mol)
1491          else:
1492              mf = scf.UHF(mol) if mol.spin != 0 else scf.RHF(mol)
1493      elif family == "DFT":
1494          mf = dft.UKS(mol) if mol.spin != 0 else dft.RKS(mol)
1495          mf.xc = meta["xc"]
1496          mf.grids.level = int(grid_level)
1497      else:
1498          raise ValueError(f"Family metode tidak dikenal: {family}")
1499
1500      if density_fitting:
1501          try:
1502              mf = mf.density_fit()
1503          except Exception:
1504              pass
1505
1506      if use_gpu:
1507          try:
1508              if hasattr(mf, "to_gpu"):
1509                  mf = mf.to_gpu()
1510                  if progress_log:
1511                      append_log(Path(progress_log), f"GPU4PySCF aktif; GPU
Threads={int(gpu_threads)}")
1512              elif progress_log:
1513                  append_log(Path(progress_log), "GPU diminta tetapi metode ini tidak
menyediakan to_gpu(); memakai CPU PySCF.")
1514          except Exception as exc:
```

```
1515              if progress_log:
1516                  append_log(Path(progress_log), f"Aktivasi GPU4PySCF gagal ({exc}); 
fallback ke CPU.")
1517
1518      mf.conv_tol = float(conv_tol)
1519      mf.max_cycle = int(max_cycle)
1520      mf.verbose = 4 if log_handle else 0
1521      if log_handle is not None:
1522          mf.stdout = log_handle
1523      if chkfile:
1524          mf.chkfile = str(chkfile)
1525
1526      def scf_callback(envs):
1527          if stop_event is not None and stop_event.is_set():
1528              raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1529          cyc = envs.get("cycle", envs.get("cycle_id", "?"))
1530          e_tot = envs.get("e_tot", envs.get("last_hf_e", None))
1531          if progress_log and e_tot is not None:
1532              append_log(progress_log, f"SCF cycle {cyc}: E = {e_tot}")
1533
1534      mf.callback = scf_callback
1535      return mf
1536
1537
1538  def make_b3lyp(mol, grid_level=3, conv_tol=1e-9, max_cycle=100):
1539      """Fungsi lama dipertahankan untuk kompatibilitas internal."""
1540      return make_pyscf_method(
1541          mol, "B3LYP", grid_level=grid_level,
1542          conv_tol=conv_tol, max_cycle=max_cycle,
1543      )
1544
1545
1546  def run_scf_with_fallback(mf, conv_tol, max_cycle, stop_event, progress_log):
1547      if stop_event.is_set():
1548          raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1549      e = float(mf.kernel())
1550      if getattr(mf, "converged", False):
1551          return mf, e
1552
1553      append_log(progress_log, "SCF biasa belum konvergen; mencoba second-order/Newton 
SCF.")
1554      newton_mf = mf.newton()
1555      newton_mf.conv_tol = float(conv_tol)
1556      newton_mf.max_cycle = int(max_cycle)
1557      newton_mf.callback = mf.callback
1558      try:
1559          newton_mf.stdout = mf.stdout
1560      except Exception:
1561          pass
1562      e = float(newton_mf.kernel())
1563      if not getattr(newton_mf, "converged", False):
1564          raise RuntimeError("SCF tidak konvergen, termasuk setelah fallback Newton.")
1565      return newton_mf, e
1566
1567
1568  def optimize_pyscf(mf, optimizer_name, maxsteps, stop_event, progress_log):
1569      if stop_event.is_set():
1570          raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1571
1572      counter = {"n": 0, "energies": []}
1573
1574      def opt_callback(envs):
1575          counter["n"] += 1
```

```
1576          if stop_event.is_set():
1577              raise StopRequested("Optimasi dihentikan oleh pengguna.")
1578          e = None
1579          try:
1580              scanner = envs.get("g_scanner")
1581              e = getattr(scanner, "e_tot", None) if scanner is not None else None
1582          except Exception:
1583              e = None
1584          if e is None:
1585              e = envs.get("energy", envs.get("e_tot"))
1586          try:
1587              e_num = float(e) if e is not None else None
1588          except Exception:
1589              e_num = None
1590          counter["energies"].append({
1591              "Step": int(counter["n"]),
1592              "Energy (Eh)": e_num,
1593              "Energy (eV)": (e_num * HARTREE_TO_EV if e_num is not None else None),
1594              "Energy (kJ/mol)": (e_num * HARTREE_TO_KJ_MOL if e_num is not None else
None),
1595              "Energy (kcal/mol)": (e_num * HARTREE_TO_KCAL_MOL if e_num is not None
else None),
1596          })
1597          append_log(progress_log, f"Geometry step {counter['n']}: E = {e}")
1598
1599      def attach_history(mol_out):
1600          try:
1601              setattr(mol_out, "_qm_opt_energy_history", list(counter["energies"]))
1602          except Exception:
1603              pass
1604          return mol_out
1605
1606      if optimizer_name.startswith("geomeTRIC"):
1607          ensure_python_package("geometric", "geometric")
1608          from pyscf.geomopt.geometric_solver import optimize as geometric_opt
1609          return attach_history(geometric_opt(mf, maxsteps=int(maxsteps),
callback=opt_callback))
1610      if optimizer_name.startswith("PyBerny"):
1611          ensure_python_package("berny", "pyberny")
1612          from pyscf.geomopt.berny_solver import optimize as berny_opt
1613          return attach_history(berny_opt(mf, maxsteps=int(maxsteps),
callback=opt_callback))
1614      if optimizer_name.startswith("ASE"):
1615          ensure_python_package("ase", "ase")
1616          # Interface resmi PySCF melalui Gradients.optimizer(solver='ase').
1617          opt = mf.nuc_grad_method().optimizer(solver="ase")
1618          try:
1619              opt.maxsteps = int(maxsteps)
1620          except Exception:
1621              pass
1622          try:
1623              opt.callback = opt_callback
1624          except Exception:
1625              pass
1626          return attach_history(opt.kernel())
1627      raise ValueError(f"Optimizer tidak dikenal: {optimizer_name}")
1628
1629
1630  def _patch_semiempirical_init_guess(mf, progress_log=None):
1631      """Compatibility shim for pyscf-semiempirical vs newer PySCF SCF API.
1632
1633      Newer PySCF kernels call ``mf.get_init_guess(..., s1e=s1e)``.  Older
1634      pyscf-semiempirical classes (RAM1/UAM1/RMINDO3/UMINDO3) expose
```

```
    ``get_init_guess(mol=None, key='minao')`` without ``s1e``/``**kwargs``.
    Patch only the *instance* so the external package itself is never modified.
    """
    try:
        sig = inspect.signature(mf.get_init_guess)
        accepts_s1e = "s1e" in sig.parameters
        accepts_kwargs = any(
            p.kind == inspect.Parameter.VAR_KEYWORD
            for p in sig.parameters.values()
        )
        if accepts_s1e or accepts_kwargs:
            return mf
    except Exception:
        # If signature inspection fails, install the harmless compatibility
        # wrapper below.  It delegates to the original bound method.
        pass

    original_get_init_guess = mf.get_init_guess

    def get_init_guess_compat(self, mol=None, key="minao", s1e=None, **kwargs):
        # ``s1e`` is intentionally ignored.  Semiempirical classes construct
        # their own minimal/orthogonalized working basis and their historical
        # initial-guess implementation does not need the overlap passed by the
        # generic PySCF SCF kernel.
        try:
            return original_get_init_guess(mol, key)
        except TypeError:
            try:
                return original_get_init_guess(mol)
            except TypeError:
                return original_get_init_guess()

    mf.get_init_guess = types.MethodType(get_init_guess_compat, mf)
    if progress_log is not None:
        append_log(
            Path(progress_log),
            f"Compatibility patch aktif untuk
{mf.__class__.__name__}.get_init_guess(s1e=...).",
        )
    return mf


def _configure_semiempirical_mf(cls, mol, config, log_handle=None,
                                stop_event=None, progress_log=None,
                                verbose=True):
    """Create and configure a semiempirical SCF object consistently."""
    mf = cls(mol)
    _patch_semiempirical_init_guess(mf, progress_log=progress_log)
    mf.conv_tol = float(config["scf_conv_tol"])
    mf.max_cycle = int(config["max_scf_cycle"])
    mf.verbose = 4 if (log_handle is not None and verbose) else 0
    if log_handle is not None:
        mf.stdout = log_handle

    def semi_scf_callback(envs):
        if stop_event is not None and stop_event.is_set():
            raise StopRequested("Perhitungan dihentikan oleh pengguna.")
        if progress_log is not None:
            cyc = envs.get("cycle", envs.get("cycle_id", None))
            e_tot = envs.get("e_tot", envs.get("last_hf_e", None))
            if cyc is not None and e_tot is not None:
                append_log(Path(progress_log), f"Semiempirical SCF cycle {cyc}: E =
{e_tot}")
```

```
1696
1697      mf.callback = semi_scf_callback
1698      return mf
1699
1700
1701  def _semiempirical_energy_for_coords(symbols, coords_angstrom, cls, charge, spin,
1702                                       config, stop_event, progress_log=None,
1703                                       log_handle=None, verbose=False):
1704      """Evaluate one AM1/MINDO3 energy at Cartesian coordinates in Angstrom."""
1705      if stop_event is not None and stop_event.is_set():
1706          raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1707
1708      atoms = [
1709          [sym, (float(x), float(y), float(z))]
1710          for sym, (x, y, z) in zip(symbols, np.asarray(coords_angstrom).reshape(-1,
3))
1711      ]
1712      mol = gto.M(
1713          atom=atoms,
1714          unit="Angstrom",
1715          charge=int(charge),
1716          spin=int(spin),
1717          verbose=0,
1718          max_memory=int(config["max_memory_mb"]),
1719      )
1720      mf = _configure_semiempirical_mf(
1721          cls, mol, config,
1722          log_handle=log_handle,
1723          stop_event=stop_event,
1724          progress_log=progress_log if verbose else None,
1725          verbose=verbose,
1726      )
1727      e = float(mf.kernel())
1728      if not getattr(mf, "converged", True):
1729          raise RuntimeError(f"{config['method']} tidak konvergen pada evaluasi
geometri numerik.")
1730      return e, mol, mf
1731
1732
1733  def _numerical_cartesian_gradient(symbols, coords_angstrom, cls, charge, spin,
1734                                    config, stop_event, progress_log=None,
1735                                    base_energy=None, step_angstrom=1.0e-3):
1736      """Forward finite-difference gradient in Eh/Angstrom.
1737
1738      AM1 closed-shell in pyscf-semiempirical currently lacks an analytical
1739      nuclear-gradient implementation.  A finite-difference fallback keeps AM1
1740      usable for gradient/optimization jobs without changing the Hamiltonian.
1741      """
1742      x = np.asarray(coords_angstrom, dtype=float).reshape(-1)
1743      if base_energy is None:
1744          base_energy, _, _ = _semiempirical_energy_for_coords(
1745              symbols, x.reshape(-1, 3), cls, charge, spin, config,
1746              stop_event, progress_log=None, verbose=False,
1747          )
1748      grad = np.zeros_like(x)
1749      h = float(step_angstrom)
1750      for idx in range(x.size):
1751          if stop_event is not None and stop_event.is_set():
1752              raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1753          xp = x.copy()
1754          xp[idx] += h
1755          ep, _, _ = _semiempirical_energy_for_coords(
1756              symbols, xp.reshape(-1, 3), cls, charge, spin, config,
```

```
1757              stop_event, progress_log=None, verbose=False,
1758          )
1759          grad[idx] = (ep - base_energy) / h
1760      if progress_log is not None:
1761          rms = float(np.sqrt(np.mean(grad ** 2)))
1762          gmax = float(np.max(np.abs(grad)))
1763          append_log(
1764              Path(progress_log),
1765              f"Numerical gradient {config['method']}: RMS={rms:.6e} Eh/Angstrom;
MAX={gmax:.6e} Eh/Angstrom; h={h:g} A",
1766          )
1767      return grad.reshape(-1, 3)
1768
1769
1770  def _optimize_semiempirical_numerical(symbols, coords0_angstrom, cls, charge, spin,
1771                                        config, stop_event, progress_log):
1772      """Fallback Cartesian L-BFGS-B optimizer using semiempirical energies.
1773
1774      Used when the selected semiempirical method does not expose a working
1775      analytical ``nuc_grad_method``.  The electronic method remains AM1/MINDO3;
1776      only the geometry gradient is evaluated numerically.
1777      """
1778      try:
1779          ensure_python_package("scipy", "scipy")
1780          from scipy.optimize import minimize
1781      except Exception as exc:
1782          raise RuntimeError(
1783              f"Optimasi numerik AM1 memerlukan SciPy dan instalasi otomatis gagal:
{exc}"
1784          ) from exc
1785
1786      progress_path = Path(progress_log)
1787      x0 = np.asarray(coords0_angstrom, dtype=float).reshape(-1)
1788      cache = {"x": None, "e": None, "grad": None, "eval": 0, "iter": 0}
1789
1790      def same_x(x):
1791          return cache["x"] is not None and np.array_equal(np.asarray(x), cache["x"])
1792
1793      def objective(x):
1794          if stop_event.is_set():
1795              raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1796          x = np.asarray(x, dtype=float)
1797          if same_x(x) and cache["e"] is not None:
1798              return float(cache["e"])
1799          e, _, _ = _semiempirical_energy_for_coords(
1800              symbols, x.reshape(-1, 3), cls, charge, spin, config,
1801              stop_event, progress_log=None, verbose=False,
1802          )
1803          cache["x"] = x.copy()
1804          cache["e"] = float(e)
1805          cache["grad"] = None
1806          cache["eval"] += 1
1807          append_log(progress_path, f"{config['method']} numerical optimizer energy
eval {cache['eval']}: E = {e:.12f} Eh")
1808          return float(e)
1809
1810      def jacobian(x):
1811          if stop_event.is_set():
1812              raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1813          x = np.asarray(x, dtype=float)
1814          e0 = objective(x)
1815          if same_x(x) and cache["grad"] is not None:
1816              return cache["grad"].copy()
```

```
1817          g = _numerical_cartesian_gradient(
1818              symbols, x.reshape(-1, 3), cls, charge, spin, config,
1819              stop_event, progress_log=progress_path,
1820              base_energy=e0, step_angstrom=1.0e-3,
1821          ).reshape(-1)
1822          cache["x"] = x.copy()
1823          cache["e"] = e0
1824          cache["grad"] = g.copy()
1825          return g
1826
1827      def opt_callback(xk):
1828          cache["iter"] += 1
1829          e = objective(xk)
1830          g = jacobian(xk)
1831          append_log(
1832              progress_path,
1833              f"{config['method']} numerical optimization step {cache['iter']}:
E={e:.12f} Eh; "
1834              f"RMSgrad={np.sqrt(np.mean(g**2)):.6e} Eh/Angstrom; "
1835              f"MAXgrad={np.max(np.abs(g)):.6e} Eh/Angstrom",
1836          )
1837          if stop_event.is_set():
1838              raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1839
1840      append_log(
1841          progress_path,
1842          f"{config['method']} analytical gradient tidak tersedia; memakai fallback
SciPy L-BFGS-B "
1843          "dengan finite-difference Cartesian (h=1e-3 Angstrom).",
1844      )
1845
1846      result = minimize(
1847          objective,
1848          x0,
1849          method="L-BFGS-B",
1850          jac=jacobian,
1851          callback=opt_callback,
1852          options={
1853              "maxiter": int(config["max_opt_steps"]),
1854              "ftol": 1.0e-10,
1855              "gtol": 1.0e-4,
1856              "maxls": 30,
1857          },
1858      )
1859      if stop_event.is_set():
1860          raise StopRequested("Perhitungan dihentikan oleh pengguna.")
1861      append_log(
1862          progress_path,
1863          f"{config['method']} numerical optimizer selesai: success={result.success};
status={result.status}; "
1864          f"nit={getattr(result, 'nit', 'n/a')}; message={result.message}",
1865      )
1866      # A non-success status can simply mean maxiter reached.  Return the best
1867      # available geometry and let the final gradient quantify stationarity.
1868      return np.asarray(result.x, dtype=float).reshape(-1, 3), result
1869
1870
1871  def run_pyscf_semiempirical(spec, config, mol_dir, stop_event, progress_log):
1872      try:
1873          ensure_python_package("pyscf.semiempirical", "pyscf[semiempirical]")
1874          import pyscf
1875          from pyscf import semiempirical
1876      except Exception as exc:
```

```
1877          raise RuntimeError(
1878              f"Ekstensi semiempirik PySCF belum tersedia dan instalasi otomatis
gagal: {exc}"
1879          ) from exc
1880
1881      rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
1882      atoms = mol_to_pyscf_atoms(rdmol)
1883      method = config["method"]
1884      if _is_frequency_job(config["job"]):
1885          raise RuntimeError(
1886              "Frequency untuk ekstensi PySCF Semiempirical AM1/MINDO/3 tidak
dipaksakan dengan Hessian palsu. "
1887              "Pilih backend MOPAC CLI untuk vibrasi semiempirik (FORCE), atau PySCF
HF/DFT/xTB/ORCA."
1888          )
1889      charge = int(spec["charge"])
1890      spin = int(spec["multiplicity"] - 1)
1891      symbols = [a[0] for a in atoms]
1892      coords0 = np.asarray([a[1] for a in atoms], dtype=float)
1893
1894      progress_path = Path(progress_log)
1895      append_log(
1896          progress_path,
1897          f"Versi backend: PySCF={getattr(pyscf, '__version__', 'unknown')}; "
1898          f"pyscf-semiempirical={getattr(semiempirical, '__version__', 'unknown')}",
1899      )
1900
1901      with open(progress_path, "a", encoding="utf-8") as logh:
1902          mol0 = gto.M(
1903              atom=atoms,
1904              unit="Angstrom",
1905              charge=charge,
1906              spin=spin,
1907              verbose=4,
1908              max_memory=int(config["max_memory_mb"]),
1909          )
1910          mol0.stdout = logh
1911
1912          if method == "MINDO/3":
1913              cls = getattr(semiempirical, "MINDO3", None)
1914              if cls is None:
1915                  try:
1916                      from pyscf.semiempirical.mindo3 import RMINDO3, UMINDO3
1917                      cls = UMINDO3 if spin != 0 else RMINDO3
1918                  except Exception:
1919                      cls = None
1920          else:
1921              cls = getattr(semiempirical, "AM1", None)
1922              if cls is None:
1923                  try:
1924                      from pyscf.semiempirical.am1 import RAM1, UAM1
1925                      cls = UAM1 if spin != 0 else RAM1
1926                  except Exception:
1927                      cls = None
1928
1929          if cls is None:
1930              raise RuntimeError(
1931                  f"Metode {method} tidak tersedia pada versi ekstensi semiempirik
yang terpasang."
1932              )
1933
1934          mf0 = _configure_semiempirical_mf(
1935              cls, mol0, config,
```

```
1936              log_handle=logh,
1937              stop_event=stop_event,
1938              progress_log=progress_path,
1939              verbose=True,
1940          )
1941          e_initial = float(mf0.kernel())
1942          if not getattr(mf0, "converged", True):
1943              raise RuntimeError(f"{method} tidak konvergen.")
1944
1945          mol_final = mol0
1946          mf_final = mf0
1947          e_final = e_initial
1948          coords = coords0.copy()
1949          numerical_fallback = False
1950          numerical_grad_ang = None
1951
1952          wants_optimization = config["job"].startswith("Geometry Optimization")
1953          wants_gradient = ("Gradient" in config["job"]) or wants_optimization
1954
1955          if wants_optimization:
1956              # First prefer the requested PySCF optimizer when an analytical
1957              # gradient exists.  AM1/RAM1 currently raises NotImplementedError.
1958              analytic_gradient_ok = True
1959              try:
1960                  test_grad_obj = mf0.nuc_grad_method()
1961                  if test_grad_obj is None:
1962                      analytic_gradient_ok = False
1963              except (NotImplementedError, AttributeError):
1964                  analytic_gradient_ok = False
1965              except Exception as exc:
1966                  append_log(progress_path, f"Pemeriksaan gradien analitik gagal:
{exc}")
1967                  analytic_gradient_ok = False
1968
1969              if analytic_gradient_ok:
1970                  append_log(
1971                      progress_path,
1972                      f"Optimasi {method} memakai {config['optimizer']} dengan gradien
analitik.",
1973                  )
1974                  mol_final = optimize_pyscf(
1975                      mf0, config["optimizer"], config["max_opt_steps"],
1976                      stop_event, progress_path,
1977                  )
1978                  mf_final = _configure_semiempirical_mf(
1979                      cls, mol_final, config,
1980                      log_handle=logh,
1981                      stop_event=stop_event,
1982                      progress_log=progress_path,
1983                      verbose=True,
1984                  )
1985                  e_final = float(mf_final.kernel())
1986                  if not getattr(mf_final, "converged", True):
1987                      raise RuntimeError(f"SCF final {method} tidak konvergen.")
1988                  coords = np.asarray(mol_final.atom_coords(unit="Angstrom"),
dtype=float)
1989              else:
1990                  numerical_fallback = True
1991                  coords, _opt_result = _optimize_semiempirical_numerical(
1992                      symbols, coords0, cls, charge, spin, config,
1993                      stop_event, progress_path,
1994                  )
1995                  e_final, mol_final, mf_final = _semiempirical_energy_for_coords(
```

```
                    symbols, coords, cls, charge, spin, config,
                    stop_event, progress_log=progress_path,
                    log_handle=logh, verbose=True,
                )

        # Final gradient: analytical when supported, otherwise finite difference.
        grad_rms = None
        grad_max = None
        if wants_gradient:
            try:
                grad = np.asarray(mf_final.nuc_grad_method().kernel(), dtype=float)
                # PySCF analytical nuclear gradients are Eh/Bohr.
                grad_rms = float(np.sqrt(np.mean(grad ** 2)))
                grad_max = float(np.max(np.abs(grad)))
            except (NotImplementedError, AttributeError):
                numerical_fallback = True
                numerical_grad_ang = _numerical_cartesian_gradient(
                    symbols, coords, cls, charge, spin, config,
                    stop_event, progress_log=progress_path,
                    base_energy=e_final, step_angstrom=1.0e-3,
                )
                # Convert Eh/Angstrom -> Eh/Bohr for consistency with the rest
                # of the application's gradient display/output.
                BOHR_TO_ANGSTROM = 0.529177210903
                grad_bohr = numerical_grad_ang * BOHR_TO_ANGSTROM
                grad_rms = float(np.sqrt(np.mean(grad_bohr ** 2)))
                grad_max = float(np.max(np.abs(grad_bohr)))
            except Exception as exc:
                append_log(progress_path, f"Gradien final tidak dapat dihitung:
{exc}")

        frontier = calculate_frontier_orbitals(mf_final)
        orbitals_df = orbital_dataframe(mf_final)
        charges_df = mulliken_dataframe(mf_final)
        try:
            dip_vec = np.asarray(mf_final.dip_moment(unit="Debye", verbose=0),
dtype=float)
            dip_total = float(np.linalg.norm(dip_vec))
        except Exception:
            dip_vec = np.array([np.nan, np.nan, np.nan])
            dip_total = None

    final_rdmol = rdkit_with_new_coords(rdmol, coords)
    formats = export_bonded_formats(final_rdmol, f"{spec['name']} {method}")
    backend_note = "PySCF semiempirical extension"
    if numerical_fallback:
        backend_note += "; numerical Cartesian gradient fallback (SciPy L-BFGS-B /
finite difference)"

    return build_result_dict(
        spec=spec, config=config, final_rdmol=final_rdmol,
        e_initial=e_initial, e_final=e_final, mf=mf_final,
        frontier=frontier, dip_vec=dip_vec, dip_total=dip_total,
        grad_rms=grad_rms, grad_max=grad_max,
        orbitals_df=orbitals_df, charges_df=charges_df,
        formats=formats, coords=coords,
        backend_note=backend_note,
    )


def build_result_dict(spec, config, final_rdmol, e_initial, e_final, mf=None,
                      frontier=None, dip_vec=None, dip_total=None,
                      grad_rms=None, grad_max=None, orbitals_df=None,
```

```
2056                        charges_df=None, formats=None, coords=None, frequencies=None,
2057                        backend_note=""):
2058      ids = molecule_identifiers(final_rdmol)
2059      units = energy_units(e_final)
2060      delta_units = energy_units(e_final - e_initial if e_initial is not None and
e_final is not None else None)
2061      if coords is None:
2062          conf = final_rdmol.GetConformer()
2063          coords = np.array([[conf.GetAtomPosition(i).x, conf.GetAtomPosition(i).y,
conf.GetAtomPosition(i).z]
2064                             for i in range(final_rdmol.GetNumAtoms())])
2065      coords_rows = []
2066      for i, atom in enumerate(final_rdmol.GetAtoms()):
2067          coords_rows.append({
2068              "Atom": i + 1, "Unsur": atom.GetSymbol(),
2069              "X (Å)": float(coords[i, 0]), "Y (Å)": float(coords[i, 1]), "Z (Å)":
float(coords[i, 2]),
2070          })
2071      if frontier is None:
2072          frontier = {"homo_index": None, "lumo_index": None, "homo_ev": None,
"lumo_ev": None, "gap_ev": None}
2073      if dip_vec is None:
2074          dip_vec = [None, None, None]
2075      if formats is None:
2076          formats = export_bonded_formats(final_rdmol, spec["name"])
2077      if frequencies is None:
2078          frequencies = []
2079
2080      result = {
2081          "name": spec["name"],
2082          "project_name": config.get("project_name", "QM_Project"),
2083          "source": spec.get("source", ""),
2084          "backend": config["program_qm"],
2085          "backend_note": backend_note,
2086          "method": config["method"],
2087          "basis": config.get("basis", "Tidak digunakan"),
2088          "job": config["job"],
2089          "optimizer": config.get("optimizer", "-"),
2090          "addon": config.get("addon", "None"),
2091          "formula": ids["formula"],
2092          "smiles": ids["smiles"],
2093          "inchi": ids["inchi"],
2094          "inchikey": ids["inchikey"],
2095          "mw": ids["mw"],
2096          "exact_mass": ids["exact_mass"],
2097          "charge": int(spec["charge"]),
2098          "multiplicity": int(spec["multiplicity"]),
2099          "spin": int(spec["multiplicity"] - 1),
2100          "natm": int(final_rdmol.GetNumAtoms()),
2101          "nelectron": int(getattr(getattr(mf, "mol", None), "nelectron", 0)) if mf is
not None else None,
2102          "nao": int(mf.mol.nao_nr()) if mf is not None and hasattr(mf, "mol") and
getattr(mf.mol, "basis", None) else None,
2103          "scf_converged": bool(getattr(mf, "converged", True)) if mf is not None else
None,
2104          "energy_initial_h": e_initial,
2105          "energy_final_h": e_final,
2106          "energy": units,
2107          "delta_e_h": None if e_initial is None or e_final is None else e_final -
e_initial,
2108          "delta_energy": delta_units,
2109          "homo_index": frontier.get("homo_index"),
2110          "lumo_index": frontier.get("lumo_index"),
```

```
2111          "homo_ev": frontier.get("homo_ev"),
2112          "lumo_ev": frontier.get("lumo_ev"),
2113          "gap_ev": frontier.get("gap_ev"),
2114          "dipole_x_d": None if dip_vec[0] is None else float(dip_vec[0]),
2115          "dipole_y_d": None if dip_vec[1] is None else float(dip_vec[1]),
2116          "dipole_z_d": None if dip_vec[2] is None else float(dip_vec[2]),
2117          "dipole_total_d": dip_total,
2118          "grad_rms_au": grad_rms,
2119          "grad_max_au": grad_max,
2120          "frequency_count": len(frequencies),
2121          "imaginary_frequency_count": sum(1 for row in frequencies if
bool(row.get("Imaginary", False))),
2122          "frequencies": frequencies,
2123          "coords": coords_rows,
2124          "orbitals": [] if orbitals_df is None else
orbitals_df.to_dict(orient="records"),
2125          "charges": [] if charges_df is None else
charges_df.to_dict(orient="records"),
2126          "xyz": formats.get("xyz", ""),
2127          "mol": formats.get("mol", ""),
2128          "sdf": formats.get("sdf", ""),
2129          "pdb": formats.get("pdb", ""),
2130      }
2131      return result
2132
2133
2134  def _is_frequency_job(job_name):
2135      return "Frequency" in str(job_name)
2136
2137
2138  def _is_optimization_job(job_name):
2139      return str(job_name).startswith("Geometry Optimization")
2140
2141
2142  def _pyscf_hessian_to_frequencies(mol, hessian):
2143      """Konversi Hessian PySCF (Eh/Bohr^2) menjadi frekuensi harmonik cm^-1."""
2144      h = np.asarray(hessian, dtype=float)
2145      n = int(mol.natm)
2146      if h.shape == (n, n, 3, 3):
2147          h = h.transpose(0, 2, 1, 3).reshape(3 * n, 3 * n)
2148      elif h.shape != (3 * n, 3 * n):
2149          h = h.reshape(3 * n, 3 * n)
2150      h = 0.5 * (h + h.T)
2151
2152      masses = np.asarray(mol.atom_mass_list(), dtype=float)
2153      m3 = np.repeat(masses, 3)
2154      mw_h = h / np.sqrt(np.outer(m3, m3))
2155      eigvals = np.linalg.eigvalsh(mw_h)
2156
2157      if n == 1:
2158          remove_n = 3
2159      elif n == 2:
2160          remove_n = 5
2161      else:
2162          coords = np.asarray(mol.atom_coords(unit="Angstrom"), dtype=float)
2163          com = np.average(coords, axis=0, weights=masses)
2164          inertia = np.zeros((3, 3), dtype=float)
2165          for mass, r in zip(masses, coords - com):
2166              inertia += mass * ((np.dot(r, r) * np.eye(3)) - np.outer(r, r))
2167          moments = np.linalg.eigvalsh(inertia)
2168          linear = bool(np.min(np.abs(moments)) < max(1.0e-8, np.max(np.abs(moments))
* 1.0e-6))
2169          remove_n = 5 if linear else 6
```

```
2170
2171      order = np.argsort(np.abs(eigvals))
2172      keep = np.ones(len(eigvals), dtype=bool)
2173      keep[order[:min(remove_n, len(eigvals))]] = False
2174      vib_vals = eigvals[keep]
2175      factor = 5140.487143715828
2176      freqs = np.sign(vib_vals) * np.sqrt(np.abs(vib_vals)) * factor
2177      return [
2178          {"Mode": i, "Frequency (cm-1)": float(freq), "Imaginary": bool(freq < 0.0)}
2179          for i, freq in enumerate(freqs, start=1)
2180      ]
2181
2182
2183  def calculate_pyscf_frequencies(mf, mol, progress_log, return_hessian=False):
2184      append_log(Path(progress_log), "Menghitung Hessian/frekuensi harmonik PySCF.")
2185      try:
2186          hess = np.asarray(mf.Hessian().kernel(), dtype=float)
2187      except Exception as exc:
2188          raise RuntimeError(f"Hessian tidak tersedia untuk metode/referensi PySCF
ini: {exc}") from exc
2189      rows = _pyscf_hessian_to_frequencies(mol, hess)
2190      append_log(Path(progress_log), f"Frequency selesai: {len(rows)} mode;
imaginary={sum(1 for r in rows if r['Imaginary'])}.")
2191      return (rows, hess) if return_hessian else rows
2192
2193
2194  def run_pyscf_calculation(spec, config, mol_dir, stop_event, progress_log):
2195      method = config["method"]
2196      if method not in PYSCF_METHODS:
2197          raise ValueError(f"Metode PySCF tidak dikenal: {method}")
2198      if config["job"].startswith("Geometry Optimization") and not
PYSCF_METHODS[method]["opt"]:
2199          raise ValueError(f"{method} pada aplikasi ini dibatasi untuk Single Point
agar hasil tidak menyesatkan.")
2200
2201      rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
2202      if rdmol is None:
2203          raise ValueError("MolBlock input gagal dibaca kembali oleh worker.")
2204      atoms = mol_to_pyscf_atoms(rdmol)
2205      charge = int(spec["charge"])
2206      spin = int(spec["multiplicity"] - 1)
2207      basis = config["basis"]
2208
2209      progress_path = Path(progress_log)
2210      chkfile = mol_dir / f"{safe_slug(spec['name'])}.chk"
2211
2212      with open(progress_path, "a", encoding="utf-8") as logh:
2213          mol0 = make_pyscf_molecule(
2214              atoms, basis=basis, charge=charge, spin=spin,
2215              max_memory_mb=config["max_memory_mb"], log_handle=logh,
2216          )
2217          mf0 = make_pyscf_method(
2218              mol0, method,
2219              grid_level=config["grid_level"],
2220              conv_tol=config["scf_conv_tol"],
2221              max_cycle=config["max_scf_cycle"],
2222              density_fitting=config["density_fitting"],
2223              chkfile=chkfile,
2224              log_handle=logh,
2225              stop_event=stop_event,
2226              progress_log=progress_path,
2227              use_gpu=bool(config.get("use_gpu", False)),
2228              gpu_threads=int(config.get("gpu_threads", 1)),
```

```
2229          )
2230
2231          append_log(progress_path, f"Mulai SCF awal {method}/{basis}")
2232          mf0, e_hf_initial = run_scf_with_fallback(
2233              mf0, config["scf_conv_tol"], config["max_scf_cycle"], stop_event,
progress_path
2234          )
2235
2236          meta = PYSCF_METHODS[method]
2237          if meta["family"] == "MP2":
2238              append_log(progress_path, "Mulai korelasi MP2 single-point.")
2239              post = mp.MP2(mf0)
2240              post.max_cycle = int(config["max_scf_cycle"])
2241              post.conv_tol = float(config["scf_conv_tol"])
2242              post.kernel()
2243              e_initial = float(post.e_tot)
2244              e_final = e_initial
2245              mol_final = mol0
2246              mf_final = mf0  # orbital/properti satu-elektron memakai reference HF
2247          else:
2248              e_initial = e_hf_initial
2249              mol_final = mol0
2250              mf_final = mf0
2251              e_final = e_initial
2252
2253              if config["job"].startswith("Geometry Optimization"):
2254                  append_log(progress_path, f"Mulai optimasi geometri dengan
{config['optimizer']}")
2255                  mol_final = optimize_pyscf(
2256                      mf0, config["optimizer"], config["max_opt_steps"], stop_event,
progress_path
2257                  )
2258                  if stop_event.is_set():
2259                      raise StopRequested("Perhitungan dihentikan oleh pengguna.")
2260                  append_log(progress_path, "Optimasi selesai; menjalankan single-
point final pada geometri akhir.")
2261                  mf_final = make_pyscf_method(
2262                      mol_final, method,
2263                      grid_level=config["grid_level"],
2264                      conv_tol=config["scf_conv_tol"],
2265                      max_cycle=config["max_scf_cycle"],
2266                      density_fitting=config["density_fitting"],
2267                      chkfile=chkfile,
2268                      log_handle=logh,
2269                      stop_event=stop_event,
2270                      progress_log=progress_path,
2271                      use_gpu=bool(config.get("use_gpu", False)),
2272                      gpu_threads=int(config.get("gpu_threads", 1)),
2273                  )
2274                  mf_final, e_final = run_scf_with_fallback(
2275                      mf_final, config["scf_conv_tol"], config["max_scf_cycle"],
stop_event, progress_path
2276                  )
2277
2278          # Properti elektronik lama dipertahankan.
2279          frontier = calculate_frontier_orbitals(mf_final)
2280          orbitals_df = orbital_dataframe(mf_final)
2281          charges_df = mulliken_dataframe(mf_final)
2282
2283          try:
2284              dip_vec = np.asarray(mf_final.dip_moment(unit="Debye", verbose=0),
dtype=float)
2285              dip_total = float(np.linalg.norm(dip_vec))
```

```
2286          except Exception:
2287              dip_vec = np.array([np.nan, np.nan, np.nan])
2288              dip_total = None
2289
2290          grad_rms = None
2291          grad_max = None
2292          if config["job"] != "Single Point" or config["job"].startswith("Geometry
Optimization"):
2293              try:
2294                  append_log(progress_path, "Menghitung gradien final.")
2295                  grad = np.asarray(mf_final.nuc_grad_method().kernel(), dtype=float)
2296                  grad_rms = float(np.sqrt(np.mean(grad ** 2)))
2297                  grad_max = float(np.max(np.abs(grad)))
2298              except Exception as exc:
2299                  append_log(progress_path, f"Gradien final tidak tersedia: {exc}")
2300
2301          frequencies = []
2302          hessian_cache = None
2303          need_hessian = _is_frequency_job(config["job"]) or config.get("addon") in
{"IR", "Thermodynamics"}
2304          if need_hessian:
2305              if meta["family"] == "MP2":
2306                  raise RuntimeError("Frequency/IR/Thermodynamics MP2 belum diaktifkan
karena Hessian MP2 tidak dihitung oleh jalur PySCF ini. Pilih HF/DFT atau backend lain.")
2307              frequencies, hessian_cache = calculate_pyscf_frequencies(mf_final,
mol_final, progress_path, return_hessian=True)
2308
2309          coords = np.asarray(mol_final.atom_coords(unit="Angstrom"), dtype=float)
2310
2311      final_rdmol = rdkit_with_new_coords(rdmol, coords)
2312      formats = export_bonded_formats(
2313          final_rdmol,
2314          f"{spec['name']} final {method}/{basis} | {config['job']} | PySCF",
2315      )
2316
2317      # v2.3.4: cube MEP/HOMO/LUMO + density dari wavefunction final.
2318      cube_files, mep_stats = generate_pyscf_cube_files(
2319          mf_final, mol_final, mol_dir, safe_slug(spec["name"]),
2320          grid_points=int(config.get("cube_grid", 60)), progress_log=progress_path,
2321      )
2322
2323      addon_meta, addon_rows, thermochemistry_rows, hessian_cache =
apply_selected_addon_pyscf(
2324          config.get("addon", "None"), mf_final, mol_final, final_rdmol, spec, config,
mol_dir,
2325          stop_event, progress_path, hessian=hessian_cache,
2326      )
2327
2328      result = build_result_dict(
2329          spec=spec, config=config, final_rdmol=final_rdmol,
2330          e_initial=e_initial, e_final=e_final, mf=mf_final,
2331          frontier=frontier, dip_vec=dip_vec, dip_total=dip_total,
2332          grad_rms=grad_rms, grad_max=grad_max,
2333          orbitals_df=orbitals_df, charges_df=charges_df,
2334          formats=formats, coords=coords, frequencies=frequencies,
2335          backend_note="Properti elektronik penuh melalui PySCF" + ("
Hessian/frekuensi harmonik dihitung" if frequencies else "") + "; MEP/HOMO/LUMO cube
v2.3.4",
2336      )
2337      result["dipole_vector_d"] = [None if not np.isfinite(x) else float(x) for x in
np.asarray(dip_vec, dtype=float)]
2338      result["dipole_magnitude_d"] = dip_total
2339      result["cube_files"] = cube_files
```

```
2340      result["mep_stats"] = mep_stats
2341      result["addon"] = config.get("addon", "None")
2342      result["addon_meta"] = addon_meta
2343      result["addon_results"] = addon_rows
2344      result["thermochemistry"] = thermochemistry_rows
2345      result["optimization_energy_history"] = list(getattr(mol_final,
"_qm_opt_energy_history", []))
2346      return result
2347
2348
2349  # ============================================================
2350  # BACKEND EKSTERNAL xTB / ORCA
2351  # ============================================================
2352  def parse_xyz_text(xyz_text):
2353      lines = [x.rstrip() for x in xyz_text.strip().splitlines()]
2354      n = int(lines[0].strip())
2355      atom_lines = lines[2:2 + n]
2356      symbols = []
2357      coords = []
2358      for line in atom_lines:
2359          p = line.split()
2360          symbols.append(p[0])
2361          coords.append([float(p[1]), float(p[2]), float(p[3])])
2362      return symbols, np.asarray(coords, dtype=float)
2363
2364
2365  def run_process_interruptible(cmd, cwd, stdout_path, stop_event, env=None):
2366      with open(stdout_path, "a", encoding="utf-8") as out:
2367          proc = subprocess.Popen(
2368              cmd, cwd=str(cwd), stdout=out, stderr=subprocess.STDOUT,
2369              text=True, env=env,
2370          )
2371          while proc.poll() is None:
2372              if stop_event.is_set():
2373                  try:
2374                      proc.terminate()
2375                      proc.wait(timeout=5)
2376                  except Exception:
2377                      try:
2378                          proc.kill()
2379                      except Exception:
2380                          pass
2381                  raise StopRequested("Proses eksternal dihentikan oleh pengguna.")
2382              time.sleep(0.5)
2383          if proc.returncode != 0:
2384              raise RuntimeError(f"Program eksternal gagal dengan exit code
{proc.returncode}. Lihat progress.log.")
2385
2386
2387  def run_xtb_calculation(spec, config, mol_dir, stop_event, progress_log):
2388      xtb = shutil.which("xtb")
2389      if not xtb:
2390          raise RuntimeError("Executable 'xtb' tidak ditemukan pada PATH.")
2391
2392      rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
2393      inp_xyz = mol_to_xyz(rdmol, spec["name"])
2394      xyz_path = mol_dir / "input.xyz"
2395      xyz_path.write_text(inp_xyz, encoding="utf-8")
2396
2397      gfn = {"GFN2-xTB": "2", "GFN1-xTB": "1", "GFN0-xTB": "0"}[config["method"]]
2398      cmd = [xtb, str(xyz_path), "--gfn", gfn, "--chrg", str(spec["charge"]), "--uhf",
str(spec["multiplicity"] - 1)]
2399      if config["job"] == "Geometry Optimization + Frequency":
```

```
2400          cmd += ["--ohess"]
2401      elif config["job"] == "Frequency Analysis":
2402          cmd += ["--hess"]
2403      elif config["job"].startswith("Geometry Optimization"):
2404          cmd += ["--opt"]
2405      append_log(progress_log, "Menjalankan: " + " ".join(cmd))
2406      run_process_interruptible(cmd, mol_dir, progress_log, stop_event)
2407
2408      log_text = Path(progress_log).read_text(encoding="utf-8", errors="replace")
2409      matches = re.findall(r"TOTAL ENERGY\s+(-?\d+\.\d+(?:[Ee][+-]?\d+)?)", log_text)
2410      if not matches:
2411          matches = re.findall(r"total energy\s+(-?\d+\.\d+(?:[Ee][+-]?\d+)?)",
log_text, flags=re.I)
2412      if not matches:
2413          raise RuntimeError("Energi xTB tidak berhasil diparsing dari progress.log.")
2414      e_final = float(matches[-1])
2415      e_initial = float(matches[0])
2416
2417      final_xyz_path = mol_dir / "xtbopt.xyz" if config["job"].startswith("Geometry
Optimization") else xyz_path
2418      if not final_xyz_path.exists():
2419          final_xyz_path = xyz_path
2420      symbols, coords = parse_xyz_text(final_xyz_path.read_text(encoding="utf-8",
errors="replace"))
2421      if len(symbols) != rdmol.GetNumAtoms():
2422          raise RuntimeError("Jumlah atom output xTB berubah/tidak cocok dengan
input.")
2423      frequencies = []
2424      if _is_frequency_job(config["job"]):
2425          freq_vals = []
2426          for line in log_text.splitlines():
2427              if re.search(r"\beigval\s*:", line, flags=re.I):
2428                  tail = line.split(":", 1)[1] if ":" in line else line
2429                  for token in re.findall(r"[-+]?\d+(?:\.\d+)?(?:[Ee][-+]?\d+)?",
tail):
2430                      try:
2431                          freq_vals.append(float(token))
2432                      except Exception:
2433                          pass
2434          # xTB prints projected frequencies including near-zero
translational/rotational modes.
2435          freq_vals = [f for f in freq_vals if abs(f) > 1.0e-2]
2436          frequencies = [
2437              {"Mode": i, "Frequency (cm-1)": float(f), "Imaginary": bool(f < 0.0)}
2438              for i, f in enumerate(freq_vals, start=1)
2439          ]
2440          append_log(Path(progress_log), f"xTB frequency parser: {len(frequencies)}
non-zero modes ditemukan.")
2441
2442      final_rdmol = rdkit_with_new_coords(rdmol, coords)
2443      formats = export_bonded_formats(final_rdmol, f"{spec['name']} {config['method']}
xTB")
2444
2445      return build_result_dict(
2446          spec=spec, config=config, final_rdmol=final_rdmol,
2447          e_initial=e_initial, e_final=e_final, mf=None,
2448          formats=formats, coords=coords, frequencies=frequencies,
2449          backend_note="xTB CLI: energi/geometri tersedia; orbital/Mulliken/dipol
mengikuti kemampuan output backend dan tidak dipaksakan." + (" Frequency melalui --hess/--
ohess." if frequencies else ""),
2450      )
2451
2452
```

```
2453  def run_orca_calculation(spec, config, mol_dir, stop_event, progress_log):
2454      orca = shutil.which("orca")
2455      if not orca:
2456          raise RuntimeError("Executable 'orca' tidak ditemukan pada PATH.")
2457
2458      rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
2459      conf = rdmol.GetConformer()
2460      method = config["method"]
2461      basis = config["basis"]
2462      if config["job"] == "Geometry Optimization + Frequency":
2463          job_keyword = "Opt Freq"
2464      elif config["job"] == "Frequency Analysis":
2465          job_keyword = "Freq"
2466      elif config["job"].startswith("Geometry Optimization"):
2467          job_keyword = "Opt"
2468      elif config["job"] == "Single Point + Gradient":
2469          job_keyword = "EnGrad"
2470      else:
2471          job_keyword = "SP"
2472      mult = int(spec["multiplicity"])
2473      charge = int(spec["charge"])
2474      nprocs = int(config["threads"])
2475      maxcore = max(100, int(config["max_memory_mb"] // max(1, nprocs)))
2476
2477      # Nama metode yang aman untuk input ORCA.
2478      method_orca = {"CAM-B3LYP": "CAM-B3LYP", "HF": "HF"}.get(method, method)
2479      lines = [
2480          f"! {method_orca} {basis} {job_keyword} TightSCF",
2481          f"%pal nprocs {nprocs} end",
2482          f"%maxcore {maxcore}",
2483          f"* xyz {charge} {mult}",
2484      ]
2485      for i, atom in enumerate(rdmol.GetAtoms()):
2486          p = conf.GetAtomPosition(i)
2487          lines.append(f"{atom.GetSymbol():2s} {p.x: .10f} {p.y: .10f} {p.z: .10f}")
2488      lines.append("*")
2489      inp = mol_dir / "orca_job.inp"
2490      inp.write_text("\n".join(lines) + "\n", encoding="utf-8")
2491
2492      append_log(progress_log, f"Menjalankan ORCA: {method}/{basis},
job={job_keyword}")
2493      run_process_interruptible([orca, str(inp)], mol_dir, progress_log, stop_event)
2494
2495      text = Path(progress_log).read_text(encoding="utf-8", errors="replace")
2496      energies = re.findall(r"FINAL SINGLE POINT ENERGY\s+(-?\d+\.\d+(?:[Ee][+-
]?\d+)?)", text)
2497      if not energies:
2498          raise RuntimeError("Energi ORCA tidak berhasil diparsing dari
progress.log.")
2499      e_final = float(energies[-1])
2500      e_initial = float(energies[0])
2501
2502      # ORCA biasanya membuat orca_job.xyz untuk optimasi.
2503      candidate_xyz = [mol_dir / "orca_job.xyz", mol_dir / "orca_job_trj.xyz"]
2504      coords = None
2505      for fp in candidate_xyz:
2506          if fp.exists() and fp.name.endswith(".xyz") and "trj" not in fp.name:
2507              try:
2508                  _, coords = parse_xyz_text(fp.read_text(encoding="utf-8",
errors="replace"))
2509                  break
2510              except Exception:
2511                  pass
```

```
2512      if coords is None:
2513          coords = np.array([[conf.GetAtomPosition(i).x, conf.GetAtomPosition(i).y,
conf.GetAtomPosition(i).z]
2514                             for i in range(rdmol.GetNumAtoms())], dtype=float)
2515
2516      frequencies = []
2517      if _is_frequency_job(config["job"]):
2518          vals = []
2519          # ORCA output umum: "  6:   1234.56 cm**-1"
2520          for m in re.finditer(r"^\s*\d+\s*:\s*(-?\d+(?:\.\d+)?)\s*cm(?:\*\*-1|\^-1|-
1)", text, flags=re.I | re.M):
2521              vals.append(float(m.group(1)))
2522          frequencies = [
2523              {"Mode": i, "Frequency (cm-1)": float(f), "Imaginary": bool(f < 0.0)}
2524              for i, f in enumerate(vals, start=1)
2525          ]
2526          append_log(Path(progress_log), f"ORCA frequency parser: {len(frequencies)}
mode ditemukan.")
2527
2528      final_rdmol = rdkit_with_new_coords(rdmol, coords)
2529      formats = export_bonded_formats(final_rdmol, f"{spec['name']} {method}/{basis}
ORCA")
2530      return build_result_dict(
2531          spec=spec, config=config, final_rdmol=final_rdmol,
2532          e_initial=e_initial, e_final=e_final, mf=None,
2533          formats=formats, coords=coords, frequencies=frequencies,
2534          backend_note="ORCA CLI: adapter generik energi/geometri. Untuk analisis
orbital lengkap gunakan backend PySCF pada aplikasi ini." + (" Frequency ORCA diparsing
dari output." if frequencies else ""),
2535      )
2536
2537
2538
2539  # ============================================================
2540  # BACKEND MOPAC CLI — semiempirik luas (RM1/PM3/PM6/PM7/dll.)
2541  # ============================================================
2542  def _mopac_spin_keywords(multiplicity):
2543      labels = {
2544          1: "SINGLET", 2: "DOUBLET", 3: "TRIPLET", 4: "QUARTET",
2545          5: "QUINTET", 6: "SEXTET", 7: "SEPTET",
2546      }
2547      mult = int(multiplicity)
2548      words = []
2549      if mult in labels:
2550          words.append(labels[mult])
2551      if mult > 1:
2552          words.append("UHF")
2553      return words
2554
2555
2556  def _write_mopac_input(path, rdmol, method, charge, multiplicity, *, optimize=False,
2557                         force=False, gradient=False, maxsteps=100, title="QM job"):
2558      keywords = [method, "AUX", "PRECISE", f"CHARGE={int(charge)}"]
2559      keywords.extend(_mopac_spin_keywords(multiplicity))
2560      if optimize:
2561          keywords += [f"CYCLES={int(maxsteps)}"]
2562      else:
2563          keywords += ["1SCF"]
2564      if force:
2565          keywords += ["FORCE", "LET"]
2566      if gradient:
2567          keywords += ["GRADIENTS"]
2568
```

```
    conf = rdmol.GetConformer()
    flag = 1 if optimize else 0
    lines = [" ".join(keywords), str(title), "Generated by app_qm_streamlit_v2_2"]
    for i, atom in enumerate(rdmol.GetAtoms()):
        pt = conf.GetAtomPosition(i)
        lines.append(
            f"{atom.GetSymbol():2s} {pt.x: .10f} {flag:d} {pt.y: .10f} {flag:d}
{pt.z: .10f} {flag:d}"
        )
    lines.append("")
    Path(path).write_text("\n".join(lines), encoding="utf-8")


def _mopac_number(text):
    """Konversi angka MOPAC, termasuk notasi Fortran D+/-xx."""
    return float(str(text).strip().replace("D", "E").replace("d", "e"))


_MOPAC_NUMBER_RE = r"[-+]?(?:\d+(?:\.\d*)?|\.\d+)(?:[EeDd][-+]?\d+)?"


def _mopac_aux_scalar(text, key):
    """Ambil scalar AUX secara toleran terhadap spasi, case, dan format MOPAC."""
    if not text:
        return None
    patterns = [
        rf"^\s*{re.escape(key)}\s*=\s*({_MOPAC_NUMBER_RE})",
        rf"^\s*{re.escape(key)}\s*:\s*({_MOPAC_NUMBER_RE})",
    ]
    for pattern in patterns:
        matches = list(re.finditer(pattern, text, flags=re.M | re.I))
        if matches:
            try:
                return _mopac_number(matches[-1].group(1))
            except Exception:
                pass
    return None


def _mopac_aux_scalar_any(text, keys):
    for key in keys:
        value = _mopac_aux_scalar(text, key)
        if value is not None:
            return value, key
    return None, None


def _mopac_coords_from_aux(text, natm):
    keys = ["ATOM_X_OPT:ANGSTROMS", "ATOM_X_UPDATED:ANGSTROMS", "ATOM_X:ANGSTROMS"]
    for key in keys:
        matches = list(re.finditer(rf"^\s*{re.escape(key)}\[\d+\]\s*=\s*$", text,
flags=re.M | re.I))
        for m in reversed(matches):
            tail = text[m.end():]
            stop = re.search(r"^\s*[A-Z][A-Z0-9_().:/+-]*(?:\[\d+\])?\s*=", tail,
flags=re.M | re.I)
            chunk = tail[:stop.start()] if stop else tail
            vals = []
            for tok in re.findall(_MOPAC_NUMBER_RE, chunk):
                try:
                    vals.append(_mopac_number(tok))
                except Exception:
                    pass
```

```
2629              if len(vals) >= 3 * natm:
2630                  return np.asarray(vals[:3 * natm], dtype=float).reshape(natm, 3)
2631      return None
2632
2633
2634  def _mopac_coords_from_out(text, natm):
2635      # Fallback: ambil tabel CARTESIAN COORDINATES terakhir yang lengkap.
2636      starts = [m.start() for m in re.finditer(r"CARTESIAN COORDINATES", text,
flags=re.I)]
2637      for start in reversed(starts):
2638          chunk = text[start:start + max(4000, natm * 160)]
2639          rows = []
2640          for line in chunk.splitlines():
2641              m = re.match(
2642                  rf"^\s*\d+\s+([A-Za-
z]{{1,3}})\s+({_MOPAC_NUMBER_RE})\s+({_MOPAC_NUMBER_RE})\s+({_MOPAC_NUMBER_RE})(?:\s+.*)?$
",
2643                  line,
2644                  flags=re.I,
2645              )
2646              if m:
2647                  try:
2648                      rows.append([_mopac_number(m.group(2)),
_mopac_number(m.group(3)), _mopac_number(m.group(4))])
2649                  except Exception:
2650                      continue
2651                  if len(rows) == natm:
2652                      return np.asarray(rows, dtype=float)
2653      return None
2654
2655
2656  def _mopac_total_energy_details(out_text, aux_text=""):
2657      """Kembalikan (energy_hartree, source, details) tanpa menyamakan HOF dengan
total energy.
2658
2659      MOPAC normalnya menulis TOTAL ENERGY dalam eV pada .out. File AUX yang lebih
2660      machine-readable lazimnya menyediakan ENERGY_ELECTRONIC:EV dan
2661      ENERGY_NUCLEAR:EV. Beberapa versi/build menggunakan nama TOTAL_ENERGY:EV.
2662      """
2663      details = {}
2664
2665      # 1) TOTAL ENERGY dari output utama: toleran terhadap '=' / ':' / whitespace.
2666      out_patterns = [
2667
rf"TOTAL\s+ENERGY\s*(?:=|:)\s*({_MOPAC_NUMBER_RE})\s*(?:EV|ELECTRON\s*VOLTS?)",
2668          rf"TOTAL\s+ENERGY\s+({_MOPAC_NUMBER_RE})\s*(?:EV|ELECTRON\s*VOLTS?)",
2669          rf"TOTAL_ENERGY(?::EV)?\s*=\s*({_MOPAC_NUMBER_RE})",
2670      ]
2671      for pattern in out_patterns:
2672          matches = list(re.finditer(pattern, out_text or "", flags=re.I | re.M))
2673          if matches:
2674              value_ev = _mopac_number(matches[-1].group(1))
2675              details["total_energy_ev"] = value_ev
2676              return value_ev / HARTREE_TO_EV, "OUT:TOTAL ENERGY", details
2677
2678      # 2) TOTAL ENERGY langsung dari AUX, bila disediakan build MOPAC tersebut.
2679      aux_total, aux_total_key = _mopac_aux_scalar_any(
2680          aux_text,
2681          ["TOTAL_ENERGY:EV", "ENERGY_TOTAL:EV", "TOTAL ENERGY:EV", "TOTAL_ENERGY"],
2682      )
2683      if aux_total is not None:
2684          details["total_energy_ev"] = aux_total
2685          return aux_total / HARTREE_TO_EV, f"AUX:{aux_total_key}", details
```

```
2686
2687      # 3) Bentuk resmi AUX: electronic + nuclear/core-core energy, keduanya eV.
2688      ee, ee_key = _mopac_aux_scalar_any(
2689          aux_text,
2690          ["ENERGY_ELECTRONIC:EV", "ELECTRONIC_ENERGY:EV", "ELECTRONIC ENERGY:EV"],
2691      )
2692      en, en_key = _mopac_aux_scalar_any(
2693          aux_text,
2694          ["ENERGY_NUCLEAR:EV", "ENERGY_CORE_CORE:EV", "CORE_CORE_REPULSION:EV",
"NUCLEAR_ENERGY:EV"],
2695      )
2696      if ee is not None:
2697          details["electronic_energy_ev"] = ee
2698      if en is not None:
2699          details["nuclear_energy_ev"] = en
2700      if ee is not None and en is not None:
2701          total_ev = ee + en
2702          details["total_energy_ev"] = total_ev
2703          return total_ev / HARTREE_TO_EV, f"AUX:{ee_key}+{en_key}", details
2704
2705      # 4) Fallback output utama: ELECTRONIC ENERGY + CORE-CORE REPULSION.
2706      def _last_out_scalar(labels):
2707          for label in labels:
2708              matches = list(re.finditer(
2709
rf"{label}\s*(?:=|:)\s*({_MOPAC_NUMBER_RE})\s*(?:EV|ELECTRON\s*VOLTS?)",
2710                  out_text or "", flags=re.I | re.M,
2711              ))
2712              if matches:
2713                  return _mopac_number(matches[-1].group(1)), label
2714          return None, None
2715
2716      ee_out, ee_label = _last_out_scalar([r"ELECTRONIC\s+ENERGY",
r"ENERGY\s+ELECTRONIC"])
2717      en_out, en_label = _last_out_scalar([
2718          r"CORE-CORE\s+REPULSION", r"CORE\s+CORE\s+REPULSION",
2719          r"NUCLEAR\s+ENERGY", r"ENERGY\s+NUCLEAR",
2720      ])
2721      if ee_out is not None:
2722          details["electronic_energy_ev"] = ee_out
2723      if en_out is not None:
2724          details["nuclear_energy_ev"] = en_out
2725      if ee_out is not None and en_out is not None:
2726          total_ev = ee_out + en_out
2727          details["total_energy_ev"] = total_ev
2728          return total_ev / HARTREE_TO_EV, f"OUT:{ee_label}+{en_label}", details
2729
2730      return None, None, details
2731
2732
2733  def _mopac_total_energy_h(out_text, aux_text=""):
2734      """Kompatibilitas fungsi lama: hanya kembalikan energi dalam Hartree atau
None."""
2735      energy_h, _source, _details = _mopac_total_energy_details(out_text, aux_text)
2736      return energy_h
2737
2738
2739  def _mopac_heat_of_formation_kcal(out_text, aux_text=""):
2740      """Ambil heat of formation sebagai properti terpisah; jangan diklaim sebagai
Hartree."""
2741      hof, _key = _mopac_aux_scalar_any(
2742          aux_text,
2743          ["HEAT_OF_FORMATION:KCAL/MOL", "HEAT_OF_FORM_UPDATED:KCAL/MOL"],
```

```
2744      )
2745      if hof is not None:
2746          return hof
2747      patterns = [
2748          rf"HEAT\s+OF\s+FORMATION\s*(?:=|:)\s*({_MOPAC_NUMBER_RE})\s*KCAL/MOL",
2749
rf"FINAL\s+HEAT\s+OF\s+FORMATION\s*(?:=|:)\s*({_MOPAC_NUMBER_RE})\s*KCAL/MOL",
2750      ]
2751      for pattern in patterns:
2752          matches = list(re.finditer(pattern, out_text or "", flags=re.I | re.M))
2753          if matches:
2754              return _mopac_number(matches[-1].group(1))
2755      return None
2756
2757
2758  def _mopac_diagnostic_excerpt(out_text, aux_text="", max_lines=28):
2759      """Ambil pesan MOPAC yang relevan agar error parser tidak menutupi error
asli."""
2760      interesting = []
2761      keywords = re.compile(
2762          r"ERROR|FATAL|UNRECOGNI[ZS]ED|NOT\s+RECOGNI[ZS]ED|FAILED|FAILURE|"
2763          r"SCF\s+FIELD\s+WAS\s+NOT|NO\s+PARAMETER|PARAMETER.*MISSING|"
2764          r"TOTAL\s+ENERGY|HEAT\s+OF\s+FORMATION|ELECTRONIC\s+ENERGY|CORE-CORE",
2765          flags=re.I,
2766      )
2767      for label, text in (("OUT", out_text), ("AUX", aux_text)):
2768          for line in (text or "").splitlines():
2769              if keywords.search(line):
2770                  interesting.append(f"[{label}] {line.strip()}")
2771      if not interesting:
2772          tail = [ln.strip() for ln in (out_text or "").splitlines() if ln.strip()][-
max_lines:]
2773          interesting = [f"[OUT] {ln}" for ln in tail]
2774      return "\n".join(interesting[-max_lines:])
2775
2776
2777  def _mopac_frequencies(out_text, aux_text=""):
2778      vals = []
2779      for m in re.finditer(r"\bFREQ\.\s*(-?\d+(?:\.\d+)?)", out_text, flags=re.I):
2780          try:
2781              vals.append(float(m.group(1)))
2782          except Exception:
2783              pass
2784      if not vals:
2785          # Beberapa versi AUX menulis vector yang mengandung FREQ pada nama field.
2786          for m in re.finditer(r"^[^\n=]*FREQ[^\n=]*\[\d+\]\s*=\s*$", aux_text,
flags=re.I | re.M):
2787              tail = aux_text[m.end():]
2788              stop = re.search(r"^\s*[A-Z][A-Z0-9_().:/+-]*(?:\[\d+\])?\s*=", tail,
flags=re.M)
2789              chunk = tail[:stop.start()] if stop else tail
2790              for tok in re.findall(r"[-+]?\d*\.?\d+(?:[EeDd][-+]?\d+)?", chunk):
2791                  try:
2792                      vals.append(_mopac_number(tok))
2793                  except Exception:
2794                      pass
2795              if vals:
2796                  break
2797      # De-duplicate sequential repeats sometimes present in verbose descriptions.
2798      cleaned = []
2799      for v in vals:
2800          if not cleaned or abs(v - cleaned[-1]) > 1.0e-8:
2801              cleaned.append(v)
```

```
2802      return [
2803          {"Mode": i, "Frequency (cm-1)": float(v), "Imaginary": bool(v < 0.0)}
2804          for i, v in enumerate(cleaned, start=1)
2805      ]
2806
2807
2808  def _run_one_mopac(mopac, input_path, mol_dir, stop_event, progress_log):
2809      append_log(Path(progress_log), "Menjalankan MOPAC: " + str(input_path.name))
2810      run_process_interruptible([mopac, str(input_path)], mol_dir, progress_log,
stop_event)
2811
2812      # MOPAC normalnya menghapus ekstensi .mop/.dat saat membentuk nama output.
2813      stem = input_path.with_suffix("")
2814      candidates_out = [
2815          Path(str(stem) + ext) for ext in (".out", ".OUT", ".Out")
2816      ]
2817      candidates_aux = [
2818          Path(str(stem) + ext) for ext in (".aux", ".AUX", ".Aux")
2819      ]
2820      out_path = next((p for p in candidates_out if p.exists()), None)
2821      aux_path = next((p for p in candidates_aux if p.exists()), None)
2822
2823      # Fallback untuk build/wrapper yang memberi variasi kapitalisasi nama file.
2824      if out_path is None:
2825          matches = sorted(mol_dir.glob(stem.name + ".*"), key=lambda p:
p.stat().st_mtime if p.exists() else 0, reverse=True)
2826          out_path = next((p for p in matches if p.suffix.lower() == ".out"), None)
2827      if aux_path is None:
2828          matches = sorted(mol_dir.glob(stem.name + ".*"), key=lambda p:
p.stat().st_mtime if p.exists() else 0, reverse=True)
2829          aux_path = next((p for p in matches if p.suffix.lower() == ".aux"), None)
2830
2831      out_text = out_path.read_text(encoding="utf-8", errors="replace") if out_path
and out_path.exists() else ""
2832      aux_text = aux_path.read_text(encoding="utf-8", errors="replace") if aux_path
and aux_path.exists() else ""
2833      if not out_text and not aux_text:
2834          raise RuntimeError(
2835              "MOPAC selesai tetapi file .out/.aux tidak ditemukan. "
2836              f"Input={input_path}; direktori={mol_dir}. Lihat progress.log untuk
stdout/stderr MOPAC."
2837          )
2838
2839      append_log(
2840          Path(progress_log),
2841          "MOPAC output ditemukan: "
2842          f"OUT={out_path.name if out_path else 'tidak ada'}; AUX={aux_path.name if
aux_path else 'tidak ada'}",
2843      )
2844      return out_text, aux_text
2845
2846
2847  def run_mopac_calculation(spec, config, mol_dir, stop_event, progress_log):
2848      mopac = find_mopac_executable()
2849      if not mopac:
2850          raise RuntimeError("Executable MOPAC tidak ditemukan pada PATH.")
2851
2852      method = config["method"]
2853      if method not in MOPAC_METHODS:
2854          raise ValueError(f"Metode MOPAC tidak dikenal: {method}")
2855
2856      rdmol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False, sanitize=False)
2857      if rdmol is None:
```

```
2858          raise ValueError("MolBlock input gagal dibaca untuk MOPAC.")
2859      natm = rdmol.GetNumAtoms()
2860      job = config["job"]
2861      optimize = _is_optimization_job(job)
2862      wants_freq = _is_frequency_job(job)
2863
2864      primary = mol_dir / "mopac_job.mop"
2865      _write_mopac_input(
2866          primary, rdmol, method, spec["charge"], spec["multiplicity"],
2867          optimize=optimize,
2868          force=(wants_freq and not optimize),
2869          gradient=(job == "Single Point + Gradient"),
2870          maxsteps=config["max_opt_steps"],
2871          title=f"{spec['name']} | {method} | {job}",
2872      )
2873      out_text, aux_text = _run_one_mopac(mopac, primary, mol_dir, stop_event,
progress_log)
2874
2875      e_final, energy_source, energy_details = _mopac_total_energy_details(out_text,
aux_text)
2876      hof_kcal = _mopac_heat_of_formation_kcal(out_text, aux_text)
2877      if e_final is None:
2878          diagnostic = _mopac_diagnostic_excerpt(out_text, aux_text)
2879          append_log(Path(progress_log), "Parser energi MOPAC gagal. Diagnostik:\n" +
diagnostic)
2880          raise RuntimeError(
2881              "Energi total MOPAC tidak berhasil diparsing. Parser sudah memeriksa
TOTAL ENERGY, "
2882              "TOTAL_ENERGY:EV, ENERGY_ELECTRONIC + ENERGY_NUCLEAR, serta ELECTRONIC
ENERGY + "
2883              "CORE-CORE REPULSION. Ini biasanya berarti job MOPAC berhenti sebelum
Final SCF results.\n\n"
2884              "Pesan MOPAC yang relevan:\n" + (diagnostic or "(tidak ada baris
diagnostik)") +
2885              "\n\nPeriksa file mopac_job.out, mopac_job.aux, dan progress.log pada
folder job."
2886          )
2887      e_initial = e_final
2888      append_log(
2889          Path(progress_log),
2890          f"Energi total MOPAC berhasil dibaca dari {energy_source}: {e_final:.12f}
Eh"
2891          + (f"; Heat of formation={hof_kcal:.8f} kcal/mol" if hof_kcal is not None
else ""),
2892      )
2893
2894      coords = _mopac_coords_from_aux(aux_text, natm)
2895      if coords is None:
2896          coords = _mopac_coords_from_out(out_text, natm)
2897      if coords is None:
2898          conf = rdmol.GetConformer()
2899          coords = np.asarray(
2900              [[conf.GetAtomPosition(i).x, conf.GetAtomPosition(i).y,
conf.GetAtomPosition(i).z] for i in range(natm)],
2901              dtype=float,
2902          )
2903
2904      frequencies = []
2905      if wants_freq and optimize:
2906          optmol = rdkit_with_new_coords(rdmol, coords)
2907          freq_inp = mol_dir / "mopac_frequency.mop"
2908          _write_mopac_input(
2909              freq_inp, optmol, method, spec["charge"], spec["multiplicity"],
```

```
2910              optimize=False, force=True, maxsteps=config["max_opt_steps"],
2911              title=f"{spec['name']} | {method} | FORCE",
2912          )
2913          freq_out, freq_aux = _run_one_mopac(mopac, freq_inp, mol_dir, stop_event,
progress_log)
2914          frequencies = _mopac_frequencies(freq_out, freq_aux)
2915      elif wants_freq:
2916          frequencies = _mopac_frequencies(out_text, aux_text)
2917
2918      if wants_freq:
2919          append_log(Path(progress_log), f"MOPAC frequency parser: {len(frequencies)}
mode ditemukan.")
2920
2921      final_rdmol = rdkit_with_new_coords(rdmol, coords)
2922      formats = export_bonded_formats(final_rdmol, f"{spec['name']} {method} MOPAC")
2923
2924      dip_total = _mopac_aux_scalar(aux_text, "DIPOLE:DEBYE")
2925      dip_vec = [None, None, None]
2926      result = build_result_dict(
2927          spec=spec, config=config, final_rdmol=final_rdmol,
2928          e_initial=e_initial, e_final=e_final, mf=None,
2929          dip_vec=dip_vec, dip_total=dip_total,
2930          formats=formats, coords=coords, frequencies=frequencies,
2931          backend_note=(
2932              "MOPAC CLI semiempirical. Energi total dibaca secara toleran dari TOTAL
ENERGY / "
2933              "TOTAL_ENERGY:EV / ENERGY_ELECTRONIC + ENERGY_NUCLEAR / ELECTRONIC
ENERGY + "
2934              "CORE-CORE REPULSION; heat of formation disimpan sebagai properti
terpisah."
2935              + (" Vibrasi dihitung dengan FORCE." if wants_freq else "")
2936          ),
2937      )
2938      result["mopac_energy_source"] = energy_source
2939      result["mopac_total_energy_ev"] = energy_details.get("total_energy_ev")
2940      result["mopac_electronic_energy_ev"] =
energy_details.get("electronic_energy_ev")
2941      result["mopac_nuclear_energy_ev"] = energy_details.get("nuclear_energy_ev")
2942      result["mopac_heat_of_formation_kcal_mol"] = hof_kcal
2943      return result
2944
2945
2946  # ============================================================
2947  # PENYIMPANAN FILE JOB
2948  # ============================================================
2949  def save_result_files(result, mol_dir: Path):
2950      base = safe_slug(result["name"])
2951      project_base = safe_slug(result.get("project_name", "QM_Project"))
2952      project_prefix = f"{project_base}__{base}"
2953
2954      # Nama file lama tetap dipertahankan agar workflow yang sudah ada tidak rusak.
2955      (mol_dir / f"{base}_final.xyz").write_text(result.get("xyz", ""), encoding="utf-
8")
2956      (mol_dir / f"{base}_final.mol").write_text(result.get("mol", ""), encoding="utf-
8")
2957      (mol_dir / f"{base}_final.sdf").write_text(result.get("sdf", ""), encoding="utf-
8")
2958      (mol_dir / f"{base}_final.pdb").write_text(result.get("pdb", ""), encoding="utf-
8")
2959      pd.DataFrame(result.get("coords", [])).to_csv(mol_dir /
f"{base}_coordinates.csv", index=False)
2960      pd.DataFrame(result.get("orbitals", [])).to_csv(mol_dir /
f"{base}_orbitals.csv", index=False)
```

```
2961      pd.DataFrame(result.get("charges", [])).to_csv(mol_dir / f"{base}_mulliken.csv",
index=False)
2962      pd.DataFrame(result.get("frequencies", [])).to_csv(mol_dir /
f"{base}_frequencies.csv", index=False)
2963      pd.DataFrame(result.get("bond_lengths", [])).to_csv(mol_dir /
f"{base}_all_bond_lengths.csv", index=False)
2964      pd.DataFrame(result.get("bond_angles", [])).to_csv(mol_dir /
f"{base}_all_bond_angles.csv", index=False)
2965      pd.DataFrame(result.get("dihedral_angles", [])).to_csv(mol_dir /
f"{base}_all_dihedral_angles.csv", index=False)
2966      pd.DataFrame(result.get("optimization_geometry_energies", [])).to_csv(mol_dir /
f"{base}_optimization_geometry_energies.csv", index=False)
2967      pd.DataFrame(result.get("optimization_energy_history", [])).to_csv(mol_dir /
f"{base}_optimization_energy_history.csv", index=False)
2968      pd.DataFrame(result.get("qsar_qspr_descriptors", [])).to_csv(mol_dir /
f"{base}_qsar_qspr_descriptors.csv", index=False)
2969      pd.DataFrame(result.get("addon_results", [])).to_csv(mol_dir /
f"{base}_addon_results.csv", index=False)
2970      pd.DataFrame(result.get("thermochemistry", [])).to_csv(mol_dir /
f"{base}_thermochemistry.csv", index=False)
2971
2972      compact = {k: v for k, v in result.items() if k not in {"xyz", "mol", "sdf",
"pdb", "coords", "orbitals", "charges"}}
2973      write_json_atomic(mol_dir / f"{base}_result.json", compact)
2974
2975      # Salinan project-prefixed membedakan input/output antarproyek tanpa menghapus
nama lama.
2976      (mol_dir / f"{project_prefix}_final.xyz").write_text(result.get("xyz", ""),
encoding="utf-8")
2977      (mol_dir / f"{project_prefix}_final.mol").write_text(result.get("mol", ""),
encoding="utf-8")
2978      (mol_dir / f"{project_prefix}_final.sdf").write_text(result.get("sdf", ""),
encoding="utf-8")
2979      (mol_dir / f"{project_prefix}_final.pdb").write_text(result.get("pdb", ""),
encoding="utf-8")
2980      pd.DataFrame(result.get("coords", [])).to_csv(mol_dir /
f"{project_prefix}_coordinates.csv", index=False)
2981      pd.DataFrame(result.get("frequencies", [])).to_csv(mol_dir /
f"{project_prefix}_frequencies.csv", index=False)
2982      pd.DataFrame(result.get("optimization_geometry_energies", [])).to_csv(mol_dir /
f"{project_prefix}_optimization_geometry_energies.csv", index=False)
2983      pd.DataFrame(result.get("optimization_energy_history", [])).to_csv(mol_dir /
f"{project_prefix}_optimization_energy_history.csv", index=False)
2984      pd.DataFrame(result.get("qsar_qspr_descriptors", [])).to_csv(mol_dir /
f"{project_prefix}_qsar_qspr_descriptors.csv", index=False)
2985      write_json_atomic(mol_dir / f"{project_prefix}_result.json", compact)
2986
2987
2988  def write_summary_log(run_dir: Path, config, results, errors, stopped=False):
2989      lines = [
2990          "QM COMPUTATION SUMMARY",
2991          "=" * 72,
2992          f"Time: {now_text()}",
2993          f"Project: {config.get('project_name', 'QM_Project')}",
2994          f"Program QM: {config['program_qm']}",
2995          f"Method: {config['method']}",
2996          f"Basis: {config.get('basis', 'Tidak digunakan')}",
2997          f"Job: {config['job']}",
2998          f"Optimizer: {config.get('optimizer', '-')}",
2999          f"Add-On: {config.get('addon', 'None')}",
3000          f"SCF tolerance: {config['scf_conv_tol']:.0e}",
3001          f"CPU threads: {config.get('threads', 1)}",
3002          f"GPU enabled: {config.get('use_gpu', False)}",
```

```
3003          f"GPU Threads: {config.get('gpu_threads', 0)}",
3004          f"Stopped by user: {stopped}",
3005          "",
3006      ]
3007      for r in results:
3008          e = r["energy"]
3009          lines += [
3010              f"[{r['name']}]",
3011              f"Formula: {r['formula']}",
3012              f"SMILES: {r['smiles']}",
3013              f"InChI: {r['inchi']}",
3014              f"Energy = {e['hartree']:.12f} Eh (a.u.)" if e["hartree"] is not None
else "Energy = n/a",
3015              f"Energy = {e['ev']:.8f} eV" if e["ev"] is not None else "",
3016              f"Energy = {e['kj_mol']:.6f} kJ/mol" if e["kj_mol"] is not None else "",
3017              f"Energy = {e['kcal_mol']:.6f} kcal/mol" if e["kcal_mol"] is not None
else "",
3018              f"HOMO/LUMO gap = {r['gap_ev']} eV",
3019              f"Dipole = {r['dipole_total_d']} D",
3020              f"RMS gradient = {r['grad_rms_au']} Eh/Bohr",
3021              f"Frequency modes = {r.get('frequency_count', 0)}; imaginary =
{r.get('imaginary_frequency_count', 0)}",
3022              "",
3023          ]
3024      if errors:
3025          lines += ["ERRORS", "-" * 72]
3026          for e in errors:
3027              lines.append(f"{e['name']}: {e['error']}")
3028      (run_dir / "summary.log").write_text("\n".join(lines) + "\n", encoding="utf-8")
3029
3030
3031  def job_worker(run_dir_str, specs, config, stop_event):
3032      run_dir = Path(run_dir_str)
3033      progress_log = run_dir / "progress.log"
3034      error_log = run_dir / "error.log"
3035      status_file = run_dir / "status.json"
3036      results_file = run_dir / "results.json"
3037      results = []
3038      errors = []
3039
3040      def status(state, message, current=0, total=None):
3041          write_json_atomic(status_file, {
3042              "state": state,
3043              "message": message,
3044              "current": current,
3045              "total": len(specs) if total is None else total,
3046              "time": now_text(),
3047          })
3048
3049      try:
3050          append_log(progress_log, f"Job dimulai. Project={config.get('project_name',
'QM_Project')}")
3051          append_log(progress_log, f"CPU threads={config.get('threads', 1)};
GPU={config.get('use_gpu', False)}; GPU Threads={config.get('gpu_threads', 0)}")
3052          status("running", "Menyiapkan perhitungan...", 0)
3053          lib.num_threads(int(config["threads"]))
3054
3055          for i, spec in enumerate(specs, start=1):
3056              if stop_event.is_set():
3057                  raise StopRequested("Perhitungan dihentikan oleh pengguna.")
3058
3059              name = spec["name"]
3060              mol_dir = run_dir / f"{i:02d}_{safe_slug(name)}"
```

```
3061              mol_dir.mkdir(parents=True, exist_ok=True)
3062              legacy_input = mol_dir / f"{safe_slug(name)}_input.mol"
3063              legacy_input.write_text(spec["molblock"] + "\n", encoding="utf-8")
3064              project_input = mol_dir / f"{safe_slug(config.get('project_name',
'QM_Project'))}__{safe_slug(name)}_input.mol"
3065              project_input.write_text(spec["molblock"] + "\n", encoding="utf-8")
3066              status("running", f"Menghitung {name}: {config['method']} /
{config.get('basis', '')}", i - 1)
3067              append_log(progress_log, f"===== MOLEKUL {i}/{len(specs)}: {name}
=====")
3068
3069              try:
3070                  if config["program_qm"] == "PySCF":
3071                      result = run_pyscf_calculation(spec, config, mol_dir,
stop_event, progress_log)
3072                  elif config["program_qm"] == "PySCF Semiempirical":
3073                      result = run_pyscf_semiempirical(spec, config, mol_dir,
stop_event, progress_log)
3074                  elif config["program_qm"] == "MOPAC CLI":
3075                      result = run_mopac_calculation(spec, config, mol_dir,
stop_event, progress_log)
3076                  elif config["program_qm"] == "xTB CLI":
3077                      result = run_xtb_calculation(spec, config, mol_dir, stop_event,
progress_log)
3078                  elif config["program_qm"] == "ORCA CLI":
3079                      result = run_orca_calculation(spec, config, mol_dir, stop_event,
progress_log)
3080                  else:
3081                      raise ValueError(f"Program QM tidak dikenal:
{config['program_qm']}")
3082                  result = enrich_result_common(result)
3083                  results.append(result)
3084                  save_result_files(result, mol_dir)
3085                  append_log(progress_log, f"Selesai: {name}; E =
{result['energy_final_h']} Eh")
3086              except StopRequested:
3087                  raise
3088              except Exception as exc:
3089                  err = {"name": name, "error": str(exc), "traceback":
traceback.format_exc()}
3090                  errors.append(err)
3091                  append_log(progress_log, f"ERROR {name}: {exc}\n{err['traceback']}")
3092                  append_log(error_log, f"ERROR {name}: {exc}\n{err['traceback']}")
3093
3094              write_json_atomic(results_file, {"results": results, "errors": errors})
3095              status("running", f"Selesai {name}", i)
3096
3097          write_summary_log(run_dir, config, results, errors, stopped=False)
3098          write_json_atomic(results_file, {"results": results, "errors": errors})
3099          state = "completed" if results else "failed"
3100          status(state, "Perhitungan selesai." if results else "Semua perhitungan
gagal.", len(specs))
3101          append_log(progress_log, "Job selesai.")
3102      except StopRequested as exc:
3103          append_log(progress_log, str(exc))
3104          write_summary_log(run_dir, config, results, errors, stopped=True)
3105          write_json_atomic(results_file, {"results": results, "errors": errors})
3106          status("stopped", str(exc), len(results))
3107      except Exception as exc:
3108          fatal_tb = traceback.format_exc()
3109          append_log(progress_log, f"FATAL: {exc}\n{fatal_tb}")
3110          append_log(error_log, f"FATAL: {exc}\n{fatal_tb}")
3111          errors.append({"name": "JOB", "error": str(exc), "traceback": fatal_tb})
```

```
3112          write_summary_log(run_dir, config, results, errors, stopped=False)
3113          write_json_atomic(results_file, {"results": results, "errors": errors})
3114          status("failed", f"Fatal: {exc}", len(results))
3115
3116
3117  # ============================================================
3118  # INPUT MOLEKUL DARI UI
3119  # ============================================================
3120  def make_spec(name, mol, source, random_seed, preopt=True, charge=None,
multiplicity=1):
3121      mol3d = prepare_molecule_3d(mol, random_seed=random_seed, preopt=preopt)
3122      ids = molecule_identifiers(mol3d)
3123      if charge is None:
3124          charge = formal_charge(mol3d)
3125      molblock = Chem.MolToMolBlock(mol3d)
3126      return {
3127          "name": name,
3128          "source": source,
3129          "molblock": molblock,
3130          "formula": ids["formula"],
3131          "smiles": ids["smiles"],
3132          "inchi": ids["inchi"],
3133          "inchikey": ids["inchikey"],
3134          "mw": ids["mw"],
3135          "charge": int(charge),
3136          "multiplicity": int(multiplicity),
3137      }
3138
3139
3140  def _split_named_smiles(line: str, idx: int):
3141      """Pisahkan Nama=SMILES tanpa merusak tanda '=' yang merupakan bagian SMILES."""
3142      # Bare SMILES yang valid (mis. CC(=O)C) harus diprioritaskan; tanda '=' di
3143      # dalam ikatan rangkap bukan separator nama.
3144      if Chem.MolFromSmiles(line) is not None:
3145          return f"SMILES-{idx}", line
3146      if "=" in line:
3147          name, ident = line.split("=", 1)
3148          name, ident = name.strip(), ident.strip()
3149          if name and ident:
3150              return name, ident
3151      return f"SMILES-{idx}", line
3152
3153
3154  def _split_named_inchi(line: str, idx: int):
3155      """Pisahkan Nama=InChI=... dengan aman karena InChI sendiri mengandung '='."""
3156      if line.startswith("InChI="):
3157          return f"InChI-{idx}", line
3158      marker = "=InChI="
3159      pos = line.find(marker)
3160      if pos > 0:
3161          name = line[:pos].strip()
3162          ident = "InChI=" + line[pos + len(marker):].strip()
3163          if name:
3164              return name, ident
3165      return f"InChI-{idx}", line
3166
3167
3168  def parse_custom_lines(text, random_seed, preopt):
3169      """Parser kompatibilitas internal dengan separator Nama=identifier."""
3170      specs = []
3171      errors = []
3172      for idx, raw in enumerate(text.splitlines(), start=1):
3173          line = raw.strip()
```

```
3174          if not line or line.startswith("#"):
3175              continue
3176          if line.startswith("InChI=") or "=InChI=" in line:
3177              name, ident = _split_named_inchi(line, idx)
3178          else:
3179              name, ident = _split_named_smiles(line, idx)
3180          try:
3181              mol = molecule_from_identifier(ident)
3182              specs.append(make_spec(name, mol, "SMILES/InChI", random_seed,
preopt=preopt))
3183          except Exception as exc:
3184              errors.append(f"Baris {idx}: {exc}")
3185      return specs, errors
3186
3187
3188  def parse_smiles_lines(text, random_seed, preopt):
3189      """Parser khusus SMILES dengan format Nama=SMILES."""
3190      specs, errors = [], []
3191      for idx, raw in enumerate(text.splitlines(), start=1):
3192          line = raw.strip()
3193          if not line or line.startswith("#"):
3194              continue
3195          name, ident = _split_named_smiles(line, idx)
3196          if ident.startswith("InChI="):
3197              errors.append(f"Baris SMILES {idx}: terdeteksi InChI. Masukkan pada
kotak InChI, bukan kotak SMILES.")
3198              continue
3199          try:
3200              mol = Chem.MolFromSmiles(ident)
3201              if mol is None:
3202                  raise ValueError(f"SMILES tidak valid: {ident}")
3203              specs.append(make_spec(name, mol, "SMILES", random_seed, preopt=preopt))
3204          except Exception as exc:
3205              errors.append(f"Baris SMILES {idx}: {exc}")
3206      return specs, errors
3207
3208
3209  def parse_inchi_lines(text, random_seed, preopt):
3210      """Parser khusus InChI dengan format Nama=InChI=... ."""
3211      specs, errors = [], []
3212      for idx, raw in enumerate(text.splitlines(), start=1):
3213          line = raw.strip()
3214          if not line or line.startswith("#"):
3215              continue
3216          name, ident = _split_named_inchi(line, idx)
3217          if not ident.startswith("InChI="):
3218              errors.append(
3219                  f"Baris InChI {idx}: gunakan format Nama=InChI=... atau InChI=...; "
3220                  "SMILES masukkan pada kotak SMILES."
3221              )
3222              continue
3223          try:
3224              mol = Chem.MolFromInchi(ident)
3225              if mol is None:
3226                  raise ValueError("InChI tidak valid")
3227              specs.append(make_spec(name, mol, "InChI", random_seed, preopt=preopt))
3228          except Exception as exc:
3229              errors.append(f"Baris InChI {idx}: {exc}")
3230      return specs, errors
3231
3232
3233  def parse_local_paths(text, random_seed, preopt):
3234      specs = []
```

```
3235      errors = []
3236      for raw in text.splitlines():
3237          path = raw.strip().strip('"')
3238          if not path:
3239              continue
3240          try:
3241              mol = molecule_from_local_path(path)
3242              name = Path(path).stem
3243              specs.append(make_spec(name, mol, f"File lokal: {path}", random_seed,
preopt=preopt))
3244          except Exception as exc:
3245              errors.append(str(exc))
3246      return specs, errors
3247
3248
3249  def load_uploaded_molecules():
3250      """Baca katalog upload persisten. Urutan file JSON = urutan tampilan terbaru
dahulu."""
3251      data = read_json_safe(UPLOAD_MOLECULES_FILE, default=[])
3252      if isinstance(data, dict):
3253          data = data.get("molecules", [])
3254      if not isinstance(data, list):
3255          return []
3256      return [x for x in data if isinstance(x, dict) and x.get("name")]
3257
3258
3259  def _persisted_source_label(source_type: str) -> str:
3260      """Label singkat sumber katalog persisten; kompatibel dengan upload v2.2.2."""
3261      labels = {
3262          "smiles": "SMILES",
3263          "inchi": "InChI",
3264          "local_file": "File lokal",
3265          "upload": "Upload",
3266          "custom": "Input",
3267      }
3268      return labels.get(str(source_type or "").lower(), "Input")
3269
3270
3271  def _unique_persisted_name(base_name: str, entries, source_type: str = "custom",
ignore_index=None):
3272      """Buat nama katalog unik tanpa menghilangkan 50 preset bawaan asli."""
3273      base = (base_name or "Molecule").strip() or "Molecule"
3274      occupied = set(MOLECULES.keys())
3275      for idx, entry in enumerate(entries):
3276          if ignore_index is not None and idx == ignore_index:
3277              continue
3278          name = str(entry.get("name", "")).strip()
3279          if name:
3280              occupied.add(name)
3281      if base not in occupied:
3282          return base
3283
3284      suffix = _persisted_source_label(source_type)
3285      candidate = f"{base} [{suffix}]"
3286      if candidate not in occupied:
3287          return candidate
3288      n = 2
3289      while f"{base} [{suffix} {n}]" in occupied:
3290          n += 1
3291      return f"{base} [{suffix} {n}]"
3292
3293
3294  def _unique_uploaded_name(base_name: str, entries):
```

```
3295      # Fungsi v2.2.2 tetap dipertahankan untuk kompatibilitas internal.
3296      return _unique_persisted_name(base_name, entries, source_type="upload")
3297
3298
3299  def _persistent_spec_digest(spec, source_type: str, source_ref: str = "", raw_bytes:
bytes | None = None):
3300      """Fingerprint stabil untuk mencegah duplikasi akibat rerun Streamlit."""
3301      if raw_bytes is not None:
3302          return hashlib.sha256(raw_bytes).hexdigest()
3303      payload = {
3304          "source_type": str(source_type or "custom"),
3305          "input_name": str(spec.get("name", "")).strip(),
3306          "smiles": str(spec.get("smiles", "")).strip(),
3307          "inchi": str(spec.get("inchi", "")).strip(),
3308          "source_ref": str(source_ref or "").strip(),
3309      }
3310      encoded = json.dumps(payload, ensure_ascii=False, sort_keys=True).encode("utf-
8")
3311      return hashlib.sha256(encoded).hexdigest()
3312
3313
3314  def persist_molecule_spec(spec, source_type: str, source_ref: str = "",
3315                            source_filename: str = "", raw_bytes: bytes | None =
None):
3316      """
3317      Simpan/updates satu molekul ke upload_molekul.json.
3318
3319      File JSON v2.2.2 sengaja tetap digunakan agar katalog upload lama langsung
terbaca.
3320      Entri baru selalu dimasukkan pada indeks 0. Jika nama input yang sama pada jenis
3321      sumber yang sama diberi struktur baru, entri lama diperbarui lalu dipindah ke
atas.
3322      """
3323      entries = load_uploaded_molecules()
3324      source_type = str(source_type or "custom").lower()
3325      requested_name = str(spec.get("name", "")).strip() or "Molecule"
3326      digest = _persistent_spec_digest(spec, source_type, source_ref=source_ref,
raw_bytes=raw_bytes)
3327
3328      # Exact match: rerun Streamlit tidak boleh menulis ulang atau mengubah urutan
katalog.
3329      for entry in entries:
3330          old_digest = str(entry.get("catalog_digest") or entry.get("sha256") or "")
3331          if old_digest and old_digest == digest:
3332              return str(entry.get("name") or requested_name), False
3333
3334      # Jika user memakai nama input yang sama untuk jenis sumber yang sama,
perlakukan
3335      # sebagai pembaruan molekul, bukan membuat tumpukan duplikat bernama serupa.
3336      replace_index = None
3337      for idx, entry in enumerate(entries):
3338          old_source_type = str(entry.get("source_type") or ("upload" if
entry.get("sha256") else "")).lower()
3339          old_input_name = str(entry.get("input_name") or entry.get("name") or
"").strip()
3340          if old_source_type == source_type and old_input_name == requested_name:
3341              replace_index = idx
3342              break
3343
3344      if replace_index is not None:
3345          saved_name = str(entries[replace_index].get("name") or requested_name)
3346      else:
3347          saved_name = _unique_persisted_name(requested_name, entries,
```

```
source_type=source_type)
3348
3349      timestamp = now_text()
3350      entry = {
3351          "name": saved_name,
3352          "input_name": requested_name,
3353          "source_type": source_type,
3354          "source_ref": str(source_ref or ""),
3355          "source_filename": str(source_filename or ""),
3356          "saved_at": timestamp,
3357          # uploaded_at tetap ditulis untuk kompatibilitas dengan kode/data v2.2.2.
3358          "uploaded_at": timestamp,
3359          "sha256": digest if raw_bytes is not None else "",
3360          "catalog_digest": digest,
3361          "category": f"Persisted {_persisted_source_label(source_type)}",
3362          "smiles": spec.get("smiles", ""),
3363          "inchi": spec.get("inchi", ""),
3364          "inchikey": spec.get("inchikey", ""),
3365          "formula": spec.get("formula", ""),
3366          "mw": spec.get("mw"),
3367          "charge": int(spec.get("charge", 0)),
3368          "spin": max(0, int(spec.get("multiplicity", 1)) - 1),
3369          "molblock": spec.get("molblock", ""),
3370      }
3371
3372      if replace_index is not None:
3373          entries.pop(replace_index)
3374      entries.insert(0, entry)
3375      write_json_atomic(UPLOAD_MOLECULES_FILE, entries)
3376      return saved_name, True
3377
3378
3379  def persist_input_specs(specs, source_type: str):
3380      """Persistenkan hasil parser SMILES/InChI/file lokal; kembalikan status
perubahan."""
3381      changed_any = False
3382      for spec in specs:
3383          source_ref = str(spec.get("source", ""))
3384          saved_name, changed = persist_molecule_spec(
3385              spec, source_type=source_type, source_ref=source_ref
3386          )
3387          spec["name"] = saved_name
3388          changed_any = changed_any or changed
3389      return specs, changed_any
3390
3391
3392  def persist_uploaded_spec(spec, source_filename: str, raw_bytes: bytes):
3393      """Simpan molekul upload ke upload_molekul.json; molekul baru selalu di indeks
0."""
3394      return persist_molecule_spec(
3395          spec, source_type="upload", source_ref=source_filename,
3396          source_filename=source_filename, raw_bytes=raw_bytes,
3397      )
3398
3399
3400  def build_molecule_catalog():
3401      """Gabungkan semua molekul persisten (terbaru di atas) + tepat 50 preset
asli."""
3402      catalog = {}
3403      for entry in load_uploaded_molecules():
3404          name = str(entry.get("name", "")).strip()
3405          if not name:
3406              continue
```

```
3407          inferred_type = str(entry.get("source_type") or ("upload" if
entry.get("source_filename") else "custom")).lower()
3408          catalog[name] = {
3409              "smiles": entry.get("smiles", ""),
3410              "inchi": entry.get("inchi", ""),
3411              "formula": entry.get("formula", ""),
3412              "charge": int(entry.get("charge", 0)),
3413              "spin": int(entry.get("spin", 0)),
3414              "category": entry.get("category", "Persisted molecule"),
3415              "molblock": entry.get("molblock", ""),
3416              "uploaded_at": entry.get("uploaded_at", entry.get("saved_at", "")),
3417              "saved_at": entry.get("saved_at", entry.get("uploaded_at", "")),
3418              "source_type": inferred_type,
3419              "source_ref": entry.get("source_ref", ""),
3420              "source_filename": entry.get("source_filename", ""),
3421              "persisted_molecule": True,
3422              # Flag lama tetap ada agar data upload v2.2.2 tetap berperilaku sama.
3423              "persisted_upload": inferred_type == "upload",
3424          }
3425      for name, meta in MOLECULES.items():
3426          catalog[name] = dict(meta)
3427      return catalog
3428
3429
3430  def _text_input_changed_for_persistence(key: str, value: str) -> bool:
3431      """True hanya sekali untuk isi text-area tertentu sampai isinya berubah lagi."""
3432      normalized = str(value or "").strip()
3433      state_key = f"_persist_input_digest_{key}"
3434      if not normalized:
3435          st.session_state[state_key] = ""
3436          return False
3437      digest = hashlib.sha256(normalized.encode("utf-8")).hexdigest()
3438      if st.session_state.get(state_key) == digest:
3439          return False
3440      st.session_state[state_key] = digest
3441      return True
3442
3443
3444  def parse_uploads(uploaded_files, random_seed, preopt):
3445      specs = []
3446      errors = []
3447      catalog_changed = False
3448      for uf in uploaded_files or []:
3449          try:
3450              raw = uf.getvalue()
3451              mol = molecule_from_bytes(uf.name, raw)
3452              spec = make_spec(Path(uf.name).stem, mol, f"Upload: {uf.name}",
random_seed, preopt=preopt)
3453              saved_name, changed = persist_uploaded_spec(spec, uf.name, raw)
3454              spec["name"] = saved_name
3455              specs.append(spec)
3456              catalog_changed = catalog_changed or changed
3457          except Exception as exc:
3458              errors.append(f"{uf.name}: {exc}")
3459      return specs, errors, catalog_changed
3460
3461
3462  def unique_names(specs):
3463      used = {}
3464      for s in specs:
3465          base = s["name"] or "molecule"
3466          n = used.get(base, 0) + 1
3467          used[base] = n
```

```
3468          if n > 1:
3469              s["name"] = f"{base}-{n}"
3470      return specs
3471
3472
3473  # ============================================================
3474  # SESSION STATE / JOB CONTROL
3475  # ============================================================
3476  def init_state():
3477      defaults = {
3478          "worker_thread": None,
3479          "worker_stop_event": None,
3480          "current_run_dir": None,
3481          "last_run_dir": None,
3482      }
3483      for k, v in defaults.items():
3484          if k not in st.session_state:
3485              st.session_state[k] = v
3486
3487
3488  def is_job_running():
3489      th = st.session_state.get("worker_thread")
3490      return bool(th is not None and th.is_alive())
3491
3492
3493  def start_background_job(specs, config):
3494      project_slug = safe_slug(config.get("project_name", "QM_Project"))
3495      run_id = project_slug + "__" + datetime.now().strftime("%Y%m%d_%H%M%S") + "_" +
uuid.uuid4().hex[:8]
3496      run_dir = RUNS_DIR / run_id
3497      run_dir.mkdir(parents=True, exist_ok=True)
3498      write_json_atomic(run_dir / "config.json", config)
3499      manifest = [{k: v for k, v in s.items() if k != "molblock"} for s in specs]
3500      write_json_atomic(run_dir / "input_manifest.json", manifest)
3501      stop_event = threading.Event()
3502      th = threading.Thread(
3503          target=job_worker,
3504          args=(str(run_dir), specs, config, stop_event),
3505          daemon=True,
3506          name=f"qm-job-{run_id}",
3507      )
3508      st.session_state.worker_stop_event = stop_event
3509      st.session_state.worker_thread = th
3510      st.session_state.current_run_dir = str(run_dir)
3511      st.session_state.last_run_dir = str(run_dir)
3512      th.start()
3513
3514
3515  def request_stop():
3516      ev = st.session_state.get("worker_stop_event")
3517      if ev is not None:
3518          ev.set()
3519
3520
3521  def copy_text_component(text, label="📋 Salin", key="copy"):
3522      """Tombol salin eksplisit dengan Clipboard API + fallback execCommand."""
3523      value_js = json.dumps(str(text), ensure_ascii=False)
3524      element_id = safe_slug(f"copy_{key}_{abs(hash(str(text))) % 10000000}")
3525      html = f"""
3526      <div style="display:flex;justify-content:flex-end;align-
items:center;height:38px;">
3527        <button id="{element_id}" style="padding:7px 12px;border:1px solid
#999;border-radius:7px;cursor:pointer;background:#fff;font-weight:600;">
```

```
3528          {label}
3529        </button>
3530        <span id="{element_id}_msg" style="margin-left:8px;font-family:sans-
serif;font-size:12px;"></span>
3531      </div>
3532      <script>
3533      const textToCopy = {value_js};
3534      const btn = document.getElementById('{element_id}');
3535      const msg = document.getElementById('{element_id}_msg');
3536      btn.addEventListener('click', async () => {{
3537        let ok = false;
3538        try {{
3539          if (navigator.clipboard && window.isSecureContext) {{
3540            await navigator.clipboard.writeText(textToCopy);
3541            ok = true;
3542          }}
3543        }} catch (e) {{}}
3544        if (!ok) {{
3545          try {{
3546            const ta = document.createElement('textarea');
3547            ta.value = textToCopy;
3548            ta.style.position = 'fixed';
3549            ta.style.left = '-9999px';
3550            document.body.appendChild(ta);
3551            ta.focus();
3552            ta.select();
3553            ok = document.execCommand('copy');
3554            document.body.removeChild(ta);
3555          }} catch (e) {{ ok = false; }}
3556        }}
3557        msg.textContent = ok ? 'Tersalin' : 'Gagal menyalin';
3558        setTimeout(() => msg.textContent = '', 1800);
3559      }});
3560      </script>
3561      """
3562      components.html(html, height=42, scrolling=False)
3563
3564
3565  # ============================================================
3566  # UI
3567  # ============================================================
3568  init_state()
3569  running = is_job_running()
3570  status_map = backend_status()
3571  gpu_info = detect_gpu_info()
3572  molecule_catalog = build_molecule_catalog()
3573
3574  st.title("⚛ Komputasi Molekul QM: PySCF / Semiempirik / MOPAC / xTB / ORCA")
3575  st.caption(
3576      "Input struktur nyata → visualisasi 2D/3D → single-point/optimasi/frekuensi →
energi multi-satuan, "
3577      "HOMO–LUMO, dipole vector/magnitude, MEP/HOMO/LUMO cube, Mulliken, gradien,
vibrasi, Add-Ons, QSAR/QSPR, log progres, dan struktur final berikatan (MOL/SDF/PDB/XYZ)."
3578  )
3579
3580  st.info(
3581      "Default tetap **PySCF + B3LYP + def2-SVP + geomeTRIC**. "
3582      "RDKit dipakai untuk konektivitas, identitas kimia, visualisasi, dan geometri
awal; "
3583      "energi/properti QM dihitung oleh backend yang dipilih."
3584  )
3585
3586  with st.sidebar:
```

```
3587      st.markdown("## QM KOMPUTASI V2.3.4")
3588      st.markdown("---")
3589      st.subheader("Input molekul")
3590      preset_selected = st.multiselect(
3591          "Molekul bawaan (50 preset + molekul tersimpan)",
3592          list(molecule_catalog.keys()),
3593          default=[],
3594          disabled=running,
3595          help=(
3596              "Secara default tidak ada molekul yang dipilih. Tepat 50 preset asli
tetap dipertahankan; "
3597              "semua molekul hasil Tambah SMILES, Tambah InChI, file lokal, dan upload
disimpan permanen "
3598              "ke upload_molekul.json lalu ditampilkan di atas preset asli, dengan
penambahan terbaru paling atas."
3599          ),
3600      )
3601      smiles_text = st.text_area(
3602          "Tambah SMILES",
3603          value="",
3604          height=100,
3605          placeholder="Satu SMILES per baris. Format Nama=SMILES.
Contoh:\nEthanol=CCO\nAcetone=CC(=O)C",
3606          disabled=running,
3607          help="Khusus SMILES. Gunakan tanda = sebagai pemisah nama dan SMILES; jangan
gunakan |.",
3608      )
3609      inchi_text = st.text_area(
3610          "Tambah InChI",
3611          value="",
3612          height=100,
3613          placeholder="Satu InChI per baris. Format Nama=InChI.
Contoh:\nWater=InChI=1S/H2O/h1H2",
3614          disabled=running,
3615          help=(
3616              "Khusus InChI. Gunakan tanda = sebagai pemisah nama dan InChI; jangan
gunakan |. "
3617              "Satu molekul cukup ditulis pada salah satu jenis input."
3618          ),
3619      )
3620      local_paths = st.text_area(
3621          "Nama/path file struktur lokal",
3622          value="",
3623          height=90,
3624          placeholder="Satu path file per baris, mis. /home/user/mol.sdf",
3625          disabled=running,
3626      )
3627      uploaded_files = st.file_uploader(
3628          "Upload struktur molekul",
3629          accept_multiple_files=True,
3630          type=None,
3631          disabled=running,
3632          help="Langsung: SDF, MOL, MOL2, PDB, XYZ, SMI/SMILES, InChI/TXT. Format lain
dicoba melalui Open Babel jika tersedia.",
3633      )
3634      st.header("Pengaturan komputasi")
3635
3636      project_name = st.text_input(
3637          "Nama proyek",
3638          value="QM_Project",
3639          disabled=running,
3640          help="Dipakai sebagai prefix folder job dan salinan nama file input/output
agar antarproyek mudah dibedakan.",
```

```
3641      ).strip() or "QM_Project"
3642
3643      program_qm = st.selectbox(
3644          "Program QM",
3645          ["PySCF", "PySCF Semiempirical", "MOPAC CLI", "xTB CLI", "ORCA CLI"],
3646          index=0,
3647          disabled=running,
3648          help="Backend selain PySCF memerlukan paket/executable terkait terpasang
pada komputer server.",
3649      )
3650
3651      job = st.selectbox("Job", JOBS, index=2, disabled=running)
3652
3653      # Override opsional dari sidebar. Default OFF agar nilai charge/multiplicity
3654      # lama dari preset, struktur, atau editor per-molekul tetap dipertahankan.
3655      sidebar_charge = st.number_input(
3656          "Charge",
3657          min_value=-20, max_value=20, value=0, step=1, disabled=running,
3658          help="Muatan total molekul. Aktifkan override di bawah agar nilai ini
diterapkan ke molekul yang akan dihitung.",
3659      )
3660      sidebar_multiplicity = st.number_input(
3661          "Multiplicity",
3662          min_value=1, max_value=20, value=1, step=1, disabled=running,
3663          help="Multiplicitas spin = 2S+1. Singlet=1, doublet=2, triplet=3, dan
seterusnya.",
3664      )
3665      use_sidebar_charge_multiplicity = st.checkbox(
3666          "Gunakan Charge & Multiplicity sidebar",
3667          value=False,
3668          disabled=running,
3669          help=(
3670              "Jika aktif, Charge dan Multiplicity di atas menjadi nilai awal untuk
semua molekul. "
3671              "Nilainya masih dapat disesuaikan lagi per molekul pada tabel editor."
3672          ),
3673      )
3674
3675      if program_qm == "PySCF":
3676          method_options = [
3677              m for m, meta in PYSCF_METHODS.items()
3678              if m not in {"RHF", "ROHF", "UHF"}
3679              and (not job.startswith("Geometry Optimization") or meta["opt"])
3680              and (not _is_frequency_job(job) or meta["family"] != "MP2")
3681          ]
3682          method_index = method_options.index("B3LYP") if "B3LYP" in method_options
else 0
3683          method_pick = st.selectbox("Metode komputasi", method_options,
index=method_index, disabled=running)
3684          if method_pick == "HF":
3685              hf_reference = st.selectbox(
3686                  "Jenis HF", ["Auto (RHF/UHF)", "RHF", "ROHF", "UHF"],
3687                  index=0, disabled=running,
3688                  help="RHF untuk closed-shell; ROHF/UHF untuk open-shell. Auto
mempertahankan perilaku lama.",
3689              )
3690              method = "HF" if hf_reference.startswith("Auto") else hf_reference
3691          else:
3692              method = method_pick
3693          basis = st.selectbox(
3694              "Basis set",
3695              BASIS_SETS,
3696              index=BASIS_SETS.index("def2-SVP"),
```

```
3697              disabled=running,
3698              help="Daftar mencakup small, medium, large, dan very large basis sets.",
3699          )
3700      elif program_qm == "PySCF Semiempirical":
3701          method = st.selectbox("Metode semiempirik PySCF", SEMIEMPIRICAL_METHODS,
index=0, disabled=running)
3702          basis = "Tidak digunakan"
3703          if _is_frequency_job(job):
3704              st.warning("Frequency AM1/MINDO/3 tidak dipaksakan pada ekstensi PySCF
semiempirik. Untuk semiempirik + frekuensi pilih MOPAC CLI.")
3705      elif program_qm == "MOPAC CLI":
3706          method = st.selectbox("Metode semiempirik MOPAC", MOPAC_METHODS,
index=MOPAC_METHODS.index("PM7"), disabled=running)
3707          basis = "Tidak digunakan"
3708          st.caption("PM4/PM5 tidak ditampilkan karena bukan Hamiltonian standar
MOPAC; PM3, RM1, PM6, PM7 dan turunannya tersedia.")
3709      elif program_qm == "xTB CLI":
3710          method = st.selectbox("Metode semiempirik xTB", XTB_METHODS, index=0,
disabled=running)
3711          basis = "Tidak digunakan"
3712      else:
3713          orca_method_options = [m for m in ORCA_METHODS if m not in {"RHF", "ROHF",
"UHF"}]
3714          method_pick = st.selectbox("Metode ORCA", orca_method_options,
index=orca_method_options.index("B3LYP"), disabled=running)
3715          if method_pick == "HF":
3716              method = st.selectbox(
3717                  "Jenis HF (ORCA)", ["RHF", "ROHF", "UHF"], index=0,
disabled=running,
3718                  help="Pilih reference Hartree–Fock secara eksplisit untuk job
ORCA.",
3719              )
3720          else:
3721              method = method_pick
3722          basis = st.selectbox("Basis set", BASIS_SETS, index=BASIS_SETS.index("def2-
SVP"), disabled=running)
3723
3724
3725      optimizer = st.selectbox(
3726          "Optimizer",
3727          OPTIMIZERS,
3728          index=0,
3729          disabled=running or not job.startswith("Geometry Optimization") or
program_qm in {"MOPAC CLI", "xTB CLI", "ORCA CLI"},
3730          help="MOPAC, xTB, dan ORCA memakai optimizer internal ketika job Geometry
Optimization dipilih.",
3731      )
3732
3733      addon = st.selectbox(
3734          "Add-Ons",
3735          ADDON_OPTIONS,
3736          index=0,
3737          disabled=running,
3738          help=(
3739              "None mempertahankan alur lama. IR, Thermodynamics, UV-Vis, dan NMR
memakai properti PySCF bila backend PySCF dipilih; "
3740              "IR hanya menghitung vibrasi harmonik, sedangkan Thermodynamics
menghitung besaran termokimia; "
3741              "Raman memakai adapter ORCA bila tersedia; ΔH f (g3mp2) memakai Gaussian
bila tersedia; "
3742              "NBO memakai ORCA + NBO6/7 dan selalu basis def2-TZPP."
3743          ),
3744      )
```

```
3745      if addon in {"Thermodynamics", "Raman"}:
3746          temperature_k = st.number_input(
3747              "Temperature (K)", min_value=1.0, max_value=5000.0, value=298.15,
step=1.0, disabled=running
3748          )
3749          pressure_pa = st.number_input(
3750              "Pressure (Pa)", min_value=1.0, max_value=100000000.0, value=101325.0,
step=1000.0, disabled=running
3751          )
3752      else:
3753          temperature_k = DEFAULT_TEMPERATURE_K
3754          pressure_pa = DEFAULT_PRESSURE_PA
3755      if addon == "UV-Vis":
3756          uvvis_nroots = st.number_input(
3757              "Jumlah excited states UV-Vis", min_value=1, max_value=200, value=20,
step=5, disabled=running
3758          )
3759      else:
3760          uvvis_nroots = 20
3761
3762      cube_grid = st.number_input(
3763          "Cube grid MEP/HOMO/LUMO", min_value=30, max_value=160, value=60, step=10,
3764          disabled=running or program_qm != "PySCF",
3765          help="Jumlah titik per sumbu untuk density/MEP/HOMO/LUMO cube. 60 = 216.000
titik per cube.",
3766      )
3767
3768      scf_conv_tol = st.selectbox(
3769          "Toleransi konvergensi SCF",
3770          options=[10.0 ** (-n) for n in range(3, 11)],
3771          index=6,  # 1e-9, mempertahankan default lama
3772          format_func=lambda x: f"{x:.0e}",
3773          disabled=running,
3774      )
3775
3776      grid_level = st.select_slider(
3777          "DFT grid level",
3778          options=[1, 2, 3, 4, 5, 6, 7, 8, 9],
3779          value=3,
3780          disabled=running or program_qm != "PySCF" or PYSCF_METHODS.get(method,
{}).get("family") != "DFT",
3781          help="Grid level terutama relevan untuk DFT PySCF.",
3782      )
3783      max_opt_steps = st.number_input(
3784          "Maksimum langkah optimasi",
3785          min_value=5, max_value=1000, value=100, step=10,
3786          disabled=running or not job.startswith("Geometry Optimization"),
3787      )
3788      max_scf_cycle = st.number_input(
3789          "Maksimum siklus SCF",
3790          min_value=20, max_value=2000, value=120, step=10,
3791          disabled=running,
3792      )
3793      threads = st.number_input(
3794          "CPU threads", min_value=1, max_value=256, value=2, step=1,
3795          disabled=running,
3796      )
3797      if gpu_info.get("detected"):
3798          st.success(f"GPU terdeteksi: {gpu_info.get('name') or 'CUDA GPU'}")
3799          gpu_threads = st.number_input(
3800              "GPU Threads", min_value=1, max_value=256, value=max(1, min(8,
int(threads))), step=1,
3801              disabled=running,
```

```
3802              help="Jumlah worker host yang dicatat untuk jalur GPU. Penjadwalan
thread/block kernel CUDA tetap dikelola driver/backend GPU.",
3803          )
3804          use_gpu = st.checkbox(
3805              "Gunakan GPU4PySCF bila tersedia",
3806              value=False,
3807              disabled=running or program_qm != "PySCF" or not
gpu_info.get("gpu4pyscf"),
3808              help="Hanya aktif bila GPU4PySCF terpasang. Jika tidak, PySCF tetap CPU
dan tidak mengubah hasil secara diam-diam.",
3809          )
3810      else:
3811          gpu_threads = 0
3812          use_gpu = False
3813      max_memory_mb = st.number_input(
3814          "Memori maksimum (MB)", min_value=256, max_value=262144, value=4000,
step=256,
3815          disabled=running,
3816      )
3817      density_fitting = st.checkbox(
3818          "Density fitting / RI (PySCF)", value=False,
3819          disabled=running or program_qm != "PySCF",
3820          help="Opsional untuk mempercepat perhitungan tertentu. Default OFF agar alur
lama tidak berubah.",
3821      )
3822      random_seed = st.number_input(
3823          "Seed geometri awal", min_value=1, max_value=2_147_483_647,
3824          value=20260820, step=1, disabled=running,
3825      )
3826      preopt = st.checkbox(
3827          "Pre-optimasi MMFF94/UFF untuk input tanpa geometri baik",
3828          value=True,
3829          disabled=running,
3830      )
3831
3832
3833      st.markdown("---")
3834      st.caption("Status backend pada komputer ini")
3835      for k, ok in status_map.items():
3836          st.write(("☑ " if ok else "⚠ ") + k)
3837      if gpu_info.get("detected"):
3838          st.caption(f"GPU: {gpu_info.get('name')} | jumlah device:
{gpu_info.get('count')} | GPU4PySCF: {'ya' if gpu_info.get('gpu4pyscf') else 'belum'}")
3839
3840      st.markdown("---")
3841      st.markdown(
3842          "<div style='text-align:center;'>Copyright @ Kasmui, 2026. All Rights
Reserved</div>",
3843          unsafe_allow_html=True,
3844      )
3845
3846  # Susun semua input molekul.
3847  input_specs = []
3848  input_errors = []
3849  for name in preset_selected:
3850      spec0 = molecule_catalog[name]
3851      try:
3852          if spec0.get("molblock"):
3853              mol0 = Chem.MolFromMolBlock(spec0["molblock"], removeHs=False,
sanitize=False)
3854              if mol0 is None:
3855                  raise ValueError("MolBlock upload tersimpan tidak dapat dibaca
kembali.")
```

```
3856              try:
3857                  Chem.SanitizeMol(mol0)
3858              except Exception:
3859                  pass
3860          else:
3861              mol0 = Chem.MolFromSmiles(spec0["smiles"])
3862          if spec0.get("persisted_molecule"):
3863              source_type0 = str(spec0.get("source_type", "custom"))
3864              source_detail0 = (
3865                  spec0.get("source_filename")
3866                  or spec0.get("source_ref")
3867                  or name
3868              )
3869              source_label = f"{_persisted_source_label(source_type0)} tersimpan:
{source_detail0}"
3870          elif spec0.get("persisted_upload"):
3871              # Cabang kompatibilitas untuk katalog lama v2.2.2 bila diperlukan.
3872              source_label = f"Upload tersimpan: {spec0.get('source_filename', name)}"
3873          else:
3874              source_label = f"Preset bawaan: {spec0.get('category', 'General')}"
3875          input_specs.append(make_spec(
3876              name, mol0, source_label, int(random_seed), preopt=preopt,
3877              charge=spec0["charge"], multiplicity=spec0["spin"] + 1,
3878          ))
3879      except Exception as exc:
3880          input_errors.append(f"{name}: {exc}")
3881
3882  specs_smiles, err_smiles = parse_smiles_lines(smiles_text, int(random_seed), preopt)
3883  specs_inchi, err_inchi = parse_inchi_lines(inchi_text, int(random_seed), preopt)
3884  specs_local, err_local = parse_local_paths(local_paths, int(random_seed), preopt)
3885
3886  # v2.3.4: semua jalur penambahan tetap disimpan permanen. Fingerprint session-state
3887  # mencegah penulisan berulang pada rerun biasa, tetapi perubahan input akan
disimpan.
3888  smiles_catalog_changed = False
3889  inchi_catalog_changed = False
3890  local_catalog_changed = False
3891  if specs_smiles and _text_input_changed_for_persistence("smiles", smiles_text):
3892      specs_smiles, smiles_catalog_changed = persist_input_specs(specs_smiles,
"smiles")
3893  if specs_inchi and _text_input_changed_for_persistence("inchi", inchi_text):
3894      specs_inchi, inchi_catalog_changed = persist_input_specs(specs_inchi, "inchi")
3895  if specs_local and _text_input_changed_for_persistence("local", local_paths):
3896      specs_local, local_catalog_changed = persist_input_specs(specs_local,
"local_file")
3897
3898  specs_upload, err_upload, upload_catalog_changed = parse_uploads(uploaded_files,
int(random_seed), preopt)
3899  catalog_changed = (
3900      smiles_catalog_changed or inchi_catalog_changed or
3901      local_catalog_changed or upload_catalog_changed
3902  )
3903  if catalog_changed:
3904      # Rerun satu kali agar molekul baru langsung muncul di posisi paling atas
multiselect.
3905      st.rerun()
3906  input_specs.extend(specs_smiles)
3907  input_specs.extend(specs_inchi)
3908  input_specs.extend(specs_local)
3909  input_specs.extend(specs_upload)
3910  input_errors.extend(err_smiles + err_inchi + err_local + err_upload)
3911  input_specs = unique_names(input_specs)
3912
```

```
3913  # Jika diminta pengguna, terapkan nilai Charge/Multiplicity dari sidebar sebagai
3914  # nilai awal. Editor per-molekul di bawah tetap dipertahankan sehingga setiap
3915  # molekul masih dapat memiliki charge/multiplicity yang berbeda.
3916  if use_sidebar_charge_multiplicity:
3917      for _spec in input_specs:
3918          _spec["charge"] = int(sidebar_charge)
3919          _spec["multiplicity"] = int(sidebar_multiplicity)
3920
3921  if input_errors:
3922      with st.expander(f"⚠ Masalah pembacaan input ({len(input_errors)})"):
3923          for e in input_errors:
3924              st.warning(e)
3925
3926  st.subheader("Daftar molekul dan identitas kimia")
3927  if input_specs:
3928      editor_df = pd.DataFrame([
3929          {
3930              "Pakai": True,
3931              "Nama": s["name"],
3932              "Formula": s["formula"],
3933              "SMILES": s["smiles"],
3934              "InChIKey": s["inchikey"],
3935              "Muatan": s["charge"],
3936              "Multiplicitas": s["multiplicity"],
3937              "Sumber": s["source"],
3938          }
3939          for s in input_specs
3940      ])
3941      edited = st.data_editor(
3942          editor_df,
3943          use_container_width=True,
3944          hide_index=True,
3945          disabled=running or ["Formula", "SMILES", "InChIKey", "Sumber"],
3946          column_config={
3947              "Pakai": st.column_config.CheckboxColumn("Pakai"),
3948              "Muatan": st.column_config.NumberColumn("Muatan", step=1),
3949              "Multiplicitas": st.column_config.NumberColumn("Multiplicitas",
min_value=1, max_value=20, step=1),
3950          },
3951          key="molecule_editor",
3952      )
3953
3954      selected_specs = []
3955      for i, row in edited.iterrows():
3956          if not bool(row["Pakai"]):
3957              continue
3958          s = dict(input_specs[i])
3959          s["name"] = str(row["Nama"]).strip() or s["name"]
3960          s["charge"] = int(row["Muatan"])
3961          s["multiplicity"] = int(row["Multiplicitas"])
3962          selected_specs.append(s)
3963  else:
3964      st.warning("Belum ada molekul. Pilih preset, masukkan SMILES atau InChI pada
kotak terpisah, path file, atau upload file struktur.")
3965      selected_specs = []
3966
3967  # Tampilkan struktur 2D/3D input.
3968  if selected_specs:
3969      st.subheader("Struktur molekul 2D / 3D")
3970      preview_tabs = st.tabs([s["name"] for s in selected_specs[:8]])
3971      for tab, spec in zip(preview_tabs, selected_specs[:8]):
3972          with tab:
3973              mol = Chem.MolFromMolBlock(spec["molblock"], removeHs=False,
```

```
sanitize=False)
3974              c2d, c3d = st.columns([1, 1.35])
3975              with c2d:
3976                  try:
3977                      img = Draw.MolToImage(Chem.RemoveHs(mol), size=(500, 330))
3978                      st.image(img, caption=f"{spec['name']} — {spec['formula']}",
use_container_width=True)
3979                  except Exception as exc:
3980                      st.info(f"2D tidak dapat dirender: {exc}")
3981                  id_df = pd.DataFrame([
3982                      ["Formula", spec["formula"]],
3983                      ["SMILES", spec["smiles"]],
3984                      ["InChI", spec["inchi"]],
3985                      ["InChIKey", spec["inchikey"]],
3986                      ["Muatan", spec["charge"]],
3987                      ["Multiplicitas", spec["multiplicity"]],
3988                  ], columns=["Identitas", "Nilai"])
3989                  st.dataframe(id_df, use_container_width=True, hide_index=True)
3990              with c3d:
3991                  html = mol_3d_html(mol)
3992                  if html:
3993                      components.html(html, height=450, scrolling=False)
3994                  else:
3995                      st.info("Visualisasi 3D interaktif memerlukan paket `py3Dmol`.
Struktur 3D tetap tersedia untuk komputasi.")
3996
3997  st.markdown("---")
3998
3999  # Snapshot konfigurasi ketika Run diklik. Perubahan UI setelahnya tidak mengubah job
yang sedang berjalan.
4000  config = {
4001      "project_name": project_name,
4002      "program_qm": program_qm,
4003      "method": method,
4004      "basis": basis,
4005      "job": job,
4006      "optimizer": optimizer,
4007      "addon": addon,
4008      "temperature_k": float(temperature_k),
4009      "pressure_pa": float(pressure_pa),
4010      "uvvis_nroots": int(uvvis_nroots),
4011      "cube_grid": int(cube_grid),
4012      "scf_conv_tol": float(scf_conv_tol),
4013      "grid_level": int(grid_level),
4014      "max_opt_steps": int(max_opt_steps),
4015      "max_scf_cycle": int(max_scf_cycle),
4016      "threads": int(threads),
4017      "gpu_threads": int(gpu_threads),
4018      "use_gpu": bool(use_gpu),
4019      "gpu_name": gpu_info.get("name", "") if gpu_info.get("detected") else "",
4020      "max_memory_mb": int(max_memory_mb),
4021      "density_fitting": bool(density_fitting),
4022      "random_seed": int(random_seed),
4023      "preopt": bool(preopt),
4024  }
4025
4026  # Tombol dibuat ringkas dan berdampingan.
4027  b1, b2, b3, b4 = st.columns([2.2, 0.8, 1.2, 1.4])
4028  run_label = f"▶ Jalankan {job}"
4029  run_button = b1.button(run_label, type="primary", disabled=running,
use_container_width=True)
4030  stop_button = b2.button("■ Stop", disabled=not running, use_container_width=True)
4031
```

```
4032  last_dir = st.session_state.get("last_run_dir")
4033  folder_feedback = None
4034  if last_dir and Path(last_dir).exists():
4035      if b3.button("🗁 Folder job", use_container_width=True):
4036          ok, detail = open_folder_os(last_dir)
4037          folder_feedback = (ok, f"Folder job dibuka: {detail}" if ok else f"Gagal
membuka folder job: {detail}")
4038      if b4.button("🗔 Buka folder", use_container_width=True):
4039          ok, detail = open_folder_os(last_dir)
4040          folder_feedback = (ok, f"Folder dibuka: {detail}" if ok else f"Gagal membuka
folder: {detail}")
4041  else:
4042      b3.button("🗁 Folder job", disabled=True, use_container_width=True)
4043      b4.button("🗔 Buka folder", disabled=True, use_container_width=True)
4044
4045  if folder_feedback:
4046      ok, text_feedback = folder_feedback
4047      if hasattr(st, "toast"):
4048          st.toast(text_feedback, icon="☑" if ok else "⚠")
4049      elif ok:
4050          st.success(text_feedback)
4051      else:
4052          st.warning(text_feedback)
4053
4054  if stop_button:
4055      request_stop()
4056      st.warning("Permintaan Stop dikirim. Job akan berhenti pada checkpoint aman
berikutnya (siklus SCF/langkah optimasi/proses eksternal).")
4057
4058  if run_button:
4059      if not selected_specs:
4060          st.error("Pilih minimal satu molekul yang valid.")
4061          st.stop()
4062      if not status_map.get(program_qm, False):
4063          if program_qm == "PySCF Semiempirical":
4064              st.info("Backend semiempirik belum terdeteksi; saat dipakai aplikasi
akan mencoba instalasi otomatis `pyscf[semiempirical]`.")
4065          elif program_qm == "MOPAC CLI":
4066              st.error("Executable MOPAC belum tersedia pada PATH. Metode
RM1/PM3/PM6/PM7 memerlukan instalasi MOPAC.")
4067          elif program_qm == "xTB CLI":
4068              st.error("Backend xTB belum tersedia pada PATH.")
4069          elif program_qm == "ORCA CLI":
4070              st.error("Executable ORCA belum tersedia pada PATH.")
4071          st.stop()
4072      start_background_job(selected_specs, config)
4073      st.rerun()
4074
4075
4076  # ============================================================
4077  # MONITOR JOB + HASIL
4078  # ============================================================
4079  def render_monitor_and_results():
4080      run_dir_text = st.session_state.get("current_run_dir") or
st.session_state.get("last_run_dir")
4081      if not run_dir_text:
4082          st.subheader("Apa yang dapat dihitung?")
4083          st.markdown(
4084              """
4085  1. Input struktur dari preset, **SMILES, InChI, file lokal, atau upload** berbagai
format kimia.
4086  2. Geometri awal 3D dengan RDKit; pre-optimasi MMFF94/UFF dapat dimatikan.
4087  3. Pilihan **HF (Auto/RHF/ROHF/UHF), DFT, MP2 single-point, semiempirik PySCF,
```

```
MOPAC, xTB**, dan backend ORCA eksternal.
4088     PySCF semiempirik mempertahankan AM1/MINDO/3; MOPAC menambah MNDO, PM3, RM1, PM6,
PM7 dan koreksinya.
4089  4. Basis set >20 pilihan dari small hingga very large; default **def2-SVP**.
4090  5. Job **Single Point**, **Gradient**, **Geometry Optimization**, **Frequency
Analysis**, dan **Geometry Optimization + Frequency**.
4091  6. Optimizer **geomeTRIC**, **PyBerny**, atau ASE (opsional) untuk PySCF; backend
eksternal memakai optimizer internal.
4092  7. Energi dalam **Hartree/Eh (a.u.), eV, kJ/mol, kcal/mol, cm⁻¹** dan frekuensi
vibrasi dalam **cm⁻¹**.
4093  8. Output final **XYZ + MOL + SDF + PDB**; MOL/SDF/PDB mempertahankan
konektivitas/ikatan dari struktur input.
4094  9. `progress.log`, `summary.log`, `config.json`, manifest input, checkpoint PySCF,
CSV koordinat/orbital/Mulliken/frekuensi.
4095  10. Nama proyek membedakan folder dan salinan nama file input/output; worker tetap
berjalan terpisah dan tombol Stop tersedia.
4096              """
4097          )
4098          return
4099
4100      run_dir = Path(run_dir_text)
4101      status = read_json_safe(run_dir / "status.json", {}) or {}
4102      payload = read_json_safe(run_dir / "results.json", {}) or {}
4103      results = payload.get("results", [])
4104      errors = payload.get("errors", [])
4105
4106      st.subheader("Status job")
4107      state = status.get("state", "starting")
4108      message = status.get("message", "Menyiapkan...")
4109      cur = int(status.get("current", 0) or 0)
4110      total = int(status.get("total", 1) or 1)
4111      st.write(f"**{state.upper()}** — {message}")
4112      progress_value = min(1.0, max(0.0, cur / max(1, total)))
4113      progress_percent = progress_value * 100.0
4114      try:
4115          st.progress(progress_value, text=f"{progress_percent:.1f}% — {cur}/{total}
molekul selesai")
4116      except TypeError:
4117          st.progress(progress_value)
4118          st.caption(f"Progress: {progress_percent:.1f}% — {cur}/{total} molekul
selesai")
4119      st.code(str(run_dir), language=None)
4120
4121      log_path = run_dir / "progress.log"
4122      if log_path.exists():
4123          try:
4124              lines = log_path.read_text(encoding="utf-8",
errors="replace").splitlines()
4125              with st.expander("📜 Progress log (baris terakhir)", expanded=state ==
"running"):
4126                  log_tail = "\n".join(lines[-120:])
4127                  lc1, lc2 = st.columns([5, 1])
4128                  with lc1:
4129                      st.code(log_tail, language=None)
4130                  with lc2:
4131                      copy_text_component(log_tail, "📋 Salin log",
key=f"progress_{run_dir.name}")
4132          except Exception:
4133              pass
4134
4135      if errors:
4136          with st.expander(f"⚠️ Error ({len(errors)})"):
4137              error_path = run_dir / "error.log"
```

```
4138              if error_path.exists():
4139                  error_text = error_path.read_text(encoding="utf-8",
errors="replace")
4140              else:
4141                  error_text = "\n\n".join(
4142                      f"[{err.get('name')}] {err.get('error')}\n{err.get('traceback',
'')}" for err in errors
4143                  )
4144              ec1, ec2 = st.columns([5, 1])
4145              with ec1:
4146                  st.code(error_text, language=None)
4147              with ec2:
4148                  copy_text_component(error_text, "📋 Salin error",
key=f"errors_{run_dir.name}")
4149              for err in errors:
4150                  st.error(f"{err.get('name')}: {err.get('error')}")
4151                  if err.get("traceback"):
4152                      st.code(err["traceback"])
4153
4154      if not results:
4155          if state == "running":
4156              st.info("Perhitungan sedang berjalan. Hasil akan muncul setelah molekul
pertama selesai.")
4157          elif state in {"failed", "stopped"}:
4158              st.warning("Belum ada hasil molekul yang selesai.")
4159          return
4160
4161      if state == "completed":
4162          st.success("Perhitungan selesai.")
4163      elif state == "stopped":
4164          st.warning("Job dihentikan. Hasil yang sudah selesai tetap disimpan.")
4165
4166      st.subheader("Ringkasan hasil")
4167      summary_rows = []
4168      for r in results:
4169          e = r["energy"]
4170          summary_rows.append({
4171              "Proyek": r.get("project_name", ""),
4172              "Molekul": r["name"],
4173              "Formula": r["formula"],
4174              "Program": r["backend"],
4175              "Metode": r["method"],
4176              "Basis": r["basis"],
4177              "Energi (Eh/a.u.)": e["hartree"],
4178              "Energi (eV)": e["ev"],
4179              "Energi (kJ/mol)": e["kj_mol"],
4180              "Gap (eV)": r.get("gap_ev"),
4181              "Dipole vector (D)": f"({r.get('dipole_x_d')}, {r.get('dipole_y_d')},
{r.get('dipole_z_d')})",
4182              "Dipole magnitude (D)": r.get("dipole_total_d"),
4183              "Add-On": r.get("addon", "None"),
4184              "RMS gradien (Eh/Bohr)": r.get("grad_rms_au"),
4185              "Mode frekuensi": r.get("frequency_count", 0),
4186              "Frekuensi imajiner": r.get("imaginary_frequency_count", 0),
4187          })
4188      summary_df = pd.DataFrame(summary_rows)
4189      st.dataframe(summary_df, use_container_width=True, hide_index=True)
4190      st.download_button(
4191          "⬇️ Unduh ringkasan CSV",
4192          data=summary_df.to_csv(index=False).encode("utf-8"),
4193          file_name="hasil_qm_summary.csv",
4194          mime="text/csv",
4195      )
```

```
4196
4197      # ZIP seluruh job dibuat setelah worker berhenti agar tidak membebani proses
yang sedang berjalan.
4198      if state != "running":
4199          try:
4200              zip_base = str(run_dir)
4201              zip_path = shutil.make_archive(zip_base, "zip", root_dir=run_dir)
4202              with open(zip_path, "rb") as f:
4203                  st.download_button(
4204                      "⬇️ Unduh seluruh input/output/log (ZIP)",
4205                      data=f.read(),
4206                      file_name=Path(zip_path).name,
4207                      mime="application/zip",
4208                  )
4209          except Exception:
4210              pass
4211
4212      for idx, r in enumerate(results):
4213          st.markdown("---")
4214          st.header(r["name"])
4215
4216          e = r["energy"]
4217          c1, c2, c3, c4 = st.columns(4)
4218          c1.metric("Energi total", f"{e['hartree']:.10f} Eh" if e["hartree"] is not
None else "n/a")
4219          c2.metric("Energi", f"{e['ev']:.6f} eV" if e["ev"] is not None else "n/a")
4220          c3.metric("HOMO", f"{r['homo_ev']:.6f} eV" if r.get("homo_ev") is not None
else "n/a")
4221          c4.metric("Gap HOMO–LUMO", f"{r['gap_ev']:.6f} eV" if r.get("gap_ev") is not
None else "n/a")
4222
4223          d1, d2, d3, d4 = st.columns(4)
4224          d1.metric("Momen dipol", f"{r['dipole_total_d']:.6f} D" if
r.get("dipole_total_d") is not None else "n/a")
4225          d2.metric("Jumlah atom", str(r["natm"]))
4226          d3.metric("Muatan", str(r["charge"]))
4227          d4.metric("Multiplicitas", str(r["multiplicity"]))
4228
4229          info_df = pd.DataFrame([
4230              ["Nama proyek", r.get("project_name", "")],
4231              ["Program QM", r["backend"]],
4232              ["Metode", r["method"]],
4233              ["Basis", r["basis"]],
4234              ["Job", r["job"]],
4235              ["Optimizer", r["optimizer"]],
4236              ["Formula", r["formula"]],
4237              ["SMILES", r["smiles"]],
4238              ["InChI", r["inchi"]],
4239              ["InChIKey", r["inchikey"]],
4240              ["Energi final (Eh/Hartree)", e["hartree"]],
4241              ["Energi final (a.u.)", e["au"]],
4242              ["Energi final (eV)", e["ev"]],
4243              ["Energi final (kJ/mol)", e["kj_mol"]],
4244              ["Energi final (kcal/mol)", e["kcal_mol"]],
4245              ["Energi final (cm⁻¹)", e["cm-1"]],
4246              ["ΔE optimasi (Eh)", r.get("delta_e_h")],
4247              ["HOMO", f"{r.get('homo_index')} = {r.get('homo_ev')} eV"],
4248              ["LUMO", f"{r.get('lumo_index')} = {r.get('lumo_ev')} eV"],
4249              ["Dipole vector (D)", f"[{r.get('dipole_x_d')}, {r.get('dipole_y_d')},
{r.get('dipole_z_d')}]"],
4250              ["Dipole magnitude (D)", r.get("dipole_total_d")],
4251              ["MEP min/max/mean (a.u.)", f"{(r.get('mep_stats') or {}).get('min')} /
{(r.get('mep_stats') or {}).get('max')} / {(r.get('mep_stats') or {}).get('mean')}"],
```

```
4252              ["Add-On", r.get("addon", "None")],
4253              ["Add-On status", (r.get("addon_meta") or {}).get("status", "")],
4254              ["RMS gradien (Eh/Bohr)", r.get("grad_rms_au")],
4255              ["Maks gradien (Eh/Bohr)", r.get("grad_max_au")],
4256              ["Jumlah mode frekuensi", r.get("frequency_count", 0)],
4257              ["Frekuensi imajiner", r.get("imaginary_frequency_count", 0)],
4258              ["Catatan backend", r.get("backend_note", "")],
4259          ], columns=["Parameter", "Nilai"])
4260          st.dataframe(info_df, use_container_width=True, hide_index=True)
4261
4262          tabs = st.tabs(["Struktur final 3D", "Koordinat", "Geometri lengkap",
"Orbital", "Muatan Mulliken", "Frekuensi", "Add-On", "QSAR/QSPR", "File struktur & cube"])
4263          with tabs[0]:
4264              mol_final = Chem.MolFromMolBlock(r.get("mol", ""), removeHs=False,
sanitize=False)
4265              if mol_final is not None:
4266                  html = mol_3d_html(mol_final)
4267                  if html:
4268                      components.html(html, height=470, scrolling=False)
4269                  try:
4270                      img = Draw.MolToImage(Chem.RemoveHs(mol_final), size=(600, 360))
4271                      st.image(img, caption="Konektivitas/ikatan final",
use_container_width=False)
4272                  except Exception:
4273                      pass
4274
4275          with tabs[1]:
4276              coords_df = pd.DataFrame(r.get("coords", []))
4277              st.dataframe(coords_df, use_container_width=True, hide_index=True)
4278
4279          with tabs[2]:
4280              st.markdown("#### Semua panjang ikatan")
4281              bond_df = pd.DataFrame(r.get("bond_lengths", []))
4282              st.dataframe(bond_df, use_container_width=True, hide_index=True)
4283              st.markdown("#### Semua sudut ikatan")
4284              angle_df = pd.DataFrame(r.get("bond_angles", []))
4285              st.dataframe(angle_df, use_container_width=True, hide_index=True)
4286              st.markdown("#### Semua sudut dihedral")
4287              dihedral_df = pd.DataFrame(r.get("dihedral_angles", []))
4288              st.dataframe(dihedral_df, use_container_width=True, hide_index=True)
4289              geom_energy_df = pd.DataFrame(r.get("optimization_geometry_energies",
[]))
4290              st.download_button(
4291                  "⬇ Unduh optimasi: geometri lengkap + semua energi CSV",
4292                  data=geom_energy_df.to_csv(index=False).encode("utf-8"),
4293
file_name=f"{safe_slug(r.get('project_name','QM_Project'))}__{safe_slug(r['name'])}_optimi
zation_geometry_energies.csv",
4294                  mime="text/csv", key=f"geom_energy_{idx}",
4295              )
4296
4297          with tabs[3]:
4298              orbitals_df = pd.DataFrame(r.get("orbitals", []))
4299              if orbitals_df.empty:
4300                  st.info("Backend ini tidak menyediakan tabel orbital lengkap melalui
adapter aplikasi.")
4301              else:
4302                  st.dataframe(orbitals_df, use_container_width=True, hide_index=True)
4303
4304          with tabs[4]:
4305              charges_df = pd.DataFrame(r.get("charges", []))
4306              if charges_df.empty:
4307                  st.info("Muatan Mulliken tidak tersedia untuk backend/metode ini
```

```
melalui adapter aplikasi.")
4308              else:
4309                  st.dataframe(charges_df, use_container_width=True, hide_index=True)
4310
4311          with tabs[5]:
4312              freq_df = pd.DataFrame(r.get("frequencies", []))
4313              if freq_df.empty:
4314                  st.info("Frekuensi vibrasi tidak dihitung untuk job/backend ini.")
4315              else:
4316                  st.dataframe(freq_df, use_container_width=True, hide_index=True)
4317                  st.download_button(
4318                      "⬇ Unduh frekuensi CSV",
4319                      data=freq_df.to_csv(index=False).encode("utf-8"),
4320
file_name=f"{safe_slug(r['project_name'])}__{safe_slug(r['name'])}_frequencies.csv",
4321                      mime="text/csv",
4322                      key=f"freq_{idx}",
4323                  )
4324                  if r.get("imaginary_frequency_count", 0):
4325                      st.warning(f"Terdapat {r.get('imaginary_frequency_count')}
frekuensi imajiner (nilai negatif). Periksa apakah geometri merupakan minimum atau keadaan
transisi.")
4326                  else:
4327                      st.success("Tidak ada frekuensi imajiner pada mode yang
terlaporkan.")
4328
4329          with tabs[6]:
4330              addon_meta = r.get("addon_meta") or {"name": r.get("addon", "None"),
"status": "n/a"}
4331              st.json(addon_meta)
4332              addon_df = pd.DataFrame(r.get("addon_results", []))
4333              thermo_df = pd.DataFrame(r.get("thermochemistry", []))
4334              if not addon_df.empty:
4335                  st.dataframe(addon_df, use_container_width=True, hide_index=True)
4336                  st.download_button(
4337                      "⬇ Unduh hasil Add-On CSV",
addon_df.to_csv(index=False).encode("utf-8"),
4338
f"{safe_slug(r.get('project_name','QM_Project'))}__{safe_slug(r['name'])}_addon_results.cs
v",
4339                      "text/csv", key=f"addon_{idx}",
4340                  )
4341              if not thermo_df.empty:
4342                  st.markdown("#### Thermochemistry")
4343                  st.dataframe(thermo_df, use_container_width=True, hide_index=True)
4344              if addon_df.empty and thermo_df.empty and r.get("addon", "None") ==
"None":
4345                  st.info("Add-On = None; tidak ada komputasi tambahan yang dipilih.")
4346
4347          with tabs[7]:
4348              desc_df = pd.DataFrame(r.get("qsar_qspr_descriptors", []))
4349              if desc_df.empty:
4350                  st.info("Deskriptor QSAR/QSPR tidak tersedia untuk hasil ini.")
4351              else:
4352                  st.caption(f"Jumlah descriptor: {len(desc_df)} (RDKit 2D/3D +
descriptor QM/conceptual-DFT bila tersedia).")
4353                  st.dataframe(desc_df, use_container_width=True, hide_index=True)
4354                  st.download_button(
4355                      "⬇ Unduh deskriptor QSAR/QSPR CSV",
desc_df.to_csv(index=False).encode("utf-8"),
4356
f"{safe_slug(r.get('project_name','QM_Project'))}__{safe_slug(r['name'])}_qsar_qspr_descri
ptors.csv",
```

```
4357                      "text/csv", key=f"qsar_{idx}",
4358                  )
4359
4360          with tabs[8]:
4361              st.caption("MOL/SDF/PDB mempertahankan tabel ikatan/konektivitas; PySCF
juga menghasilkan density/MEP/HOMO/LUMO Gaussian cube.")
4362              fc1, fc2, fc3, fc4 = st.columns(4)
4363              base = safe_slug(r["name"])
4364              project_base = safe_slug(r.get("project_name", "QM_Project"))
4365              download_base = f"{project_base}__{base}"
4366              fc1.download_button("⬇ XYZ", r.get("xyz", "").encode("utf-8"),
f"{download_base}_final.xyz", "chemical/x-xyz", key=f"xyz_{idx}")
4367              fc2.download_button("⬇ MOL", r.get("mol", "").encode("utf-8"),
f"{download_base}_final.mol", "chemical/x-mdl-molfile", key=f"mol_{idx}")
4368              fc3.download_button("⬇ SDF", r.get("sdf", "").encode("utf-8"),
f"{download_base}_final.sdf", "chemical/x-mdl-sdfile", key=f"sdf_{idx}")
4369              fc4.download_button("⬇ PDB", r.get("pdb", "").encode("utf-8"),
f"{download_base}_final.pdb", "chemical/x-pdb", key=f"pdb_{idx}")
4370              cube_files = r.get("cube_files") or {}
4371              if cube_files:
4372                  st.markdown("#### Gaussian cube")
4373                  cube_cols = st.columns(min(4, max(1, len(cube_files))))
4374                  for ci, (label, path_text) in enumerate(cube_files.items()):
4375                      p = Path(path_text)
4376                      if p.exists():
4377                          with open(p, "rb") as cf:
4378                              cube_cols[ci % len(cube_cols)].download_button(
4379                                  f"⬇ {label}", data=cf.read(), file_name=p.name,
mime="text/plain", key=f"cube_{idx}_{ci}"
4380                              )
4381
4382      # Saat worker selesai, lakukan satu full rerun agar sidebar dan tombol Run
kembali aktif.
4383      if state in {"completed", "failed", "stopped"} and not is_job_running():
4384          marker = str(run_dir) + ":" + state
4385          if st.session_state.get("_finished_job_marker") != marker:
4386              st.session_state["_finished_job_marker"] = marker
4387              st.rerun()
4388
4389
4390  # Auto-refresh monitor bila Streamlit mendukung fragments.
4391  if hasattr(st, "fragment"):
4392      _fragment_interval = 1.0 if is_job_running() else None
4393      _fragment =
st.fragment(run_every=_fragment_interval)(render_monitor_and_results)
4394      _fragment()
4395  else:
4396      render_monitor_and_results()
4397      if is_job_running():
4398          st.info("Versi Streamlit ini belum mendukung auto-refresh fragment. Klik
tombol di bawah untuk memperbarui status tanpa menghentikan worker.")
4399          if st.button("🔄 Refresh status"):
4400              st.rerun()
4401
4402  st.caption(
4403      "Catatan: tombol Stop menghentikan pada checkpoint aman. SCF PySCF diperiksa per
iterasi, "
4404      "optimasi diperiksa per langkah, dan proses MOPAC/xTB/ORCA dapat diterminasi
langsung."
4405  )
```

LAMPIRAN C — RUMUS, SATUAN, DAN CHECKLIST CEPAT

Rumus ditulis dengan karakter Unicode dan font Cambria Math, bukan sintaks LaTeX mentah. Pembaca harus memeriksa definisi, satuan, dan konteks sebelum menggunakannya.

Multiplicitas

$$\mathrm{M} = 2\mathrm{S} + 1$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Konversi Hartree ke eV

$$\mathrm{E(eV)} = \mathrm{E(E_h)} \times 27{,}211386245988$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Konversi Hartree ke kJ mol⁻¹

$$\mathrm{E(kJ\ mol^{-1})} = \mathrm{E(E_h)} \times 2625{,}4996394799$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Konversi Hartree ke kcal mol⁻¹

$$\mathrm{E(kcal\ mol^{-1})} = \mathrm{E(E_h)} \times 627{,}5094740631$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Konversi energi foton

$$\lambda(\mathrm{nm}) = 1239{,}8419843320025 \,/\, \mathrm{E(eV)}$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Jumlah mode nonlinier

$$\mathrm{N\_mode} = 3\mathrm{N} - 6$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Jumlah mode linier

$$\mathrm{N\_mode} = 3\mathrm{N} - 5$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Energi bebas Gibbs

$$\mathrm{G} = \mathrm{H} - \mathrm{TS}$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Energi nol titik harmonik

$$\mathrm{ZPE} = \tfrac{1}{2}\,\Sigma_i\, h\nu_i$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Gap frontier

$$\Delta E\_gap = E\_LUMO - E\_HOMO$$

Gunakan nilai dengan satuan konsisten dan laporkan angka signifikan sesuai ketidakpastian metode. Konstanta konversi mengikuti nilai yang tertulis dalam kode sumber aplikasi, sedangkan rumus konseptual mengikuti definisi standar kimia fisik.

Checklist sebelum menekan Run

- Identitas dan stereokimia benar.
- Charge dan multiplicity konsisten.
- Tujuan job jelas.
- Metode/basis/backend tersedia.
- Toleransi, grid, memori, thread, dan seed dicatat.
- Folder data dapat ditulis tanpa hak root.

Checklist setelah job selesai

- Log dan status konvergensi diperiksa.
- Struktur akhir masuk akal.
- Frekuensi sesuai karakter titik stasioner.
- Satuan dan konversi diverifikasi.
- Hasil sensitif diuji terhadap model.
- Folder job diarsipkan bersama versi perangkat lunak.

LAMPIRAN D — TUTORIAL QM KOMPUTASI V2.3.4

1. Download file qm-komputasi.zip ini dan letakkan di “Desktop Windows” saja
2. Ekstrak di Desktop maka akan diperoleh folder “qm-komputasi”, berisi **1 file py** (app_qm_streamlit_v2.3.4.deb) dan **1 file json (upload_molekul.json)**
3. Klik kanan di folder “qm-komputasi” untuk masuk terminal, lalu di terminal ketik “wsl” untuk masuk wsl/ubuntu
4. Setelah masul wsl ketik perintah untuk install file deb, yaitu: “sudo apt install ./app_qm_streamlit_v2.3.4.deb”
5. Apabila muncul masalah dependensi maka ketik di wsl:
   “sudo apt --fix-broken install”
   kemudian ulangi ketik di wsl:

   “sudo apt install ./app_qm_streamlit_v2.3.4.deb”
6. Pastikan paket sudah terinstal dengan cara jalankan di wsl, ketik: “dpkg -l | grep app-qm-streamlit”
7. Jika benar maka akan muncul teks "app-qm-streamlit-v2.3.4"
8. Kembali ke ***explorer Windows***: copy file “upload_molekul.json” dari folder “qm-komputasi” di Desktop
9. Lihat di explorer Windows, di sidebar kiri, ada tulisan "Linux",
10. Lalu di bawahnya ada kata "Ubuntu",
11. Lalu masuk ke "home", di situ ada nama pengguna misal "kasmui",
12. Lalu cari dan klik folder: ".local" (ada tanda titik didepan teks local),
13. Lalu cari dan klik folder "share",
14. Lalu cari dan klik teks “app-qm-streamlit-v2.3.4” atau "qm-komputasi",
15. Lalu tempelkan file “upload_molekul.json” yang sudah di-copy tadi dengan **Ctrl V**.
16. Sekarang menjalankan aplikasi di wsl dengan ketik: “app-qm-streamlit-v2.3.4”, lalu ENTER

    maka akan muncul teks berikut:

    ===

    ```
    You can now view your Streamlit app in
    your browser.

    Local URL: http://localhost:8501

    Network URL: http://172.31.159.93:8501
    ```

```
Detected WSL. Using poll-based file
watching for better compatibility. To
force watchdog, set server.fileWatcherType
= "watchdog".

===
```

17. Buka browser Windows, misal Chrome dan masukkan alamat berikut: “http://localhost:8501”
18. Buka tab baru dan masukkan link berikut juga: http://172.31.159.93:8501
19. Tunggu sebentar sampai muncul tampilan aplikasi:

20. Jika sudah muncul tampilan seperti di atas berarti sudah berfungsi.
21. Silahkan bereksplorasi.
22. Sebenarnya ini hanya bentuk visual dari bentuk “*command line”* Python kemarin.


+ HCl
computation
computing
- KASMUI -
Dosen Kimia, Komputasi, IT, AI UNNES
https://hisabmu.com/
https://kasmui.cloud/
Ketua PCM Gunungpati 2
Anggota Majelis Tabligh PDM Kota Semarang & PWM Jawa Tengah
Tim Pengembang Software KHGT MTT PP Muhammadiyah
Praktisi Ilmu Falak