PYTHON UNTUK KIMIA KUANTUM: PANDUAN PEMROGRAMAN KOMPUTASI ILMIAH

Buku pendamping orisinal tentang Python, HPC, dan aplikasi kimia kuantum

AUGUST 1, 2026

JATEN & SIBLINGS
Jaten 84 Patemon

PYTHON UNTUK KIMIA KUANTUM

Panduan Pemrograman Komputasi Ilmiah

BUKU PENDAMPING ORISINAL

Ekosistem Python • Komputasi Kinerja Tinggi • Aplikasi Kimia Kuantum

2026

PYTHON UNTUK KIMIA KUANTUM

Panduan Pemrograman Komputasi Ilmiah

Buku pendamping yang ditulis secara mandiri untuk pembelajaran, pengajaran, dan pengembangan perangkat lunak kimia komputasi.

KETERANGAN PENERBITAN DAN HAK CIPTA

Buku ini merupakan karya pendamping orisinal dalam bahasa Indonesia. Naskah disusun secara mandiri untuk menjelaskan keterampilan pemrograman Python, komputasi ilmiah, komputasi kinerja tinggi, dan penerapannya dalam kimia kuantum. Buku ini bukan terjemahan resmi, bukan reproduksi, dan bukan pengganti buku Python for Quantum Chemistry: A Full Stack Programming Guide karya Qiming Sun yang diterbitkan Elsevier.

Susunan tema tiga bagian dan enam belas bab digunakan sebagai peta pembelajaran agar pembaca dapat membandingkan cakupan bidang secara sistematis. Seluruh uraian, contoh, latihan, tabel, dan kode dalam buku ini ditulis ulang secara independen. Nama produk, pustaka, dan perangkat lunak tetap merupakan milik pemegang hak masing-masing.

Kode program diberikan untuk tujuan pendidikan. Pembaca wajib memeriksa versi pustaka, dokumentasi resmi, lisensi, kebutuhan sumber daya, konvergensi, stabilitas numerik, dan validitas metode sebelum menggunakannya untuk penelitian atau produksi.

Sitasi struktur inspirasi: Q. Sun, Python for Quantum Chemistry: A Full Stack Programming Guide, Theoretical and Computational Chemistry, Vol. 23, Elsevier, 2025.

PRAKATA

Python telah menjadi bahasa penghubung antara teori, algoritma, data, dan infrastruktur komputasi. Dalam kimia kuantum, satu program dapat memadukan model molekul, integral Gaussian, aljabar linear, tensor, proses paralel, akselerator GPU, penyimpanan data, dan layanan jaringan. Karena itu, kemampuan menulis sintaks Python saja belum cukup. Pengembang perlu memahami bagaimana seluruh lapisan tersebut bekerja bersama dan di mana biaya serta risiko kesalahan muncul.

Buku ini diarahkan kepada mahasiswa, dosen, peneliti, dan pengembang perangkat lunak yang telah memiliki dasar Python dan konsep struktur elektronik. Pembahasan tidak dimulai dari variabel dan loop dasar, serta tidak menggantikan buku teori kimia kuantum. Penekanan diberikan pada desain program, struktur data, reproducibility, pengukuran kinerja, dan penerapan teknik komputasi modern pada contoh kimia kuantum.

Setiap bab memuat capaian pembelajaran, uraian konsep, contoh kode, catatan praktik baik, ringkasan, latihan, dan proyek mini. Kode sengaja dibuat ringkas agar pola desain terlihat jelas. Pada pekerjaan nyata, tambahkan validasi input, penanganan kesalahan, logging, pengujian, dan dokumentasi.

CARA MENGGUNAKAN BUKU

Lintasan	Bab yang disarankan	Tujuan
Pengembang paket ilmiah	1–7	Membangun proyek, mengolah data, membuat visualisasi, dan mengelola I/O serta cloud.
Optimasi dan HPC	8–11	Menghubungkan bahasa, melakukan profiling, paralelisme CPU, dan akselerasi GPU.
Pengembang metode elektronik	12–16	Menerapkan integral, SCF, FCI, coupled cluster, dan sifat molekuler.
Praktikum cepat	Bab terkait + proyek mini	Mengambil contoh kode, menjalankan, memvalidasi, lalu memperluas secara bertahap.

Pembaca dapat memulai dari bagian ketiga bila fokus utama adalah implementasi metode kimia kuantum, kemudian kembali ke bagian pertama atau kedua ketika memerlukan teknik tertentu. Namun, untuk pengembangan perangkat lunak yang akan dipakai bersama, Bab 1, 2, 6, dan 9 sangat dianjurkan.

DAFTAR AKRONIM

Akronim	Kepanjangan
AD	Automatic Differentiation
AO	Atomic Orbital
API	Application Programming Interface
AST	Abstract Syntax Tree
BLAS	Basic Linear Algebra Subprograms
CC	Coupled Cluster
CCD	Coupled Cluster Doubles
CI	Configuration Interaction
CLI	Command-Line Interface
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DFT	Density Functional Theory
DIIS	Direct Inversion in the Iterative Subspace
ERI	Electron-Repulsion Integral
FFT	Fast Fourier Transform
GIL	Global Interpreter Lock
GPU	Graphics Processing Unit
GTO	Gaussian-Type Orbital
HDF5	Hierarchical Data Format 5
HF	Hartree-Fock
HPC	High-Performance Computing
I/O	Input/Output
IPC	Inter-Process Communication
JIT	Just-in-Time
JVP	Jacobian-Vector Product
MPI	Message Passing Interface
MO	Molecular Orbital
MRO	Method Resolution Order
NUMA	Non-Uniform Memory Access
RHF	Restricted Hartree-Fock

Akronim	Kepanjangan
RI	Resolution of Identity
RPC	Remote Procedure Call
SCF	Self-Consistent Field
SIMD	Single Instruction, Multiple Data
SPMD	Single Program, Multiple Data
VJP	Vector-Jacobian Product

DAFTAR ISI

BAGIAN 1

EKOSISTEM DAN PERANGKAT PYTHON UNTUK KOMPUTASI ILMIAH

Bagian ini menata lingkungan, data, visualisasi, alat numerik, metaprogramming, I/O, dan cloud sebagai fondasi perangkat lunak ilmiah yang dapat direproduksi.

BAB 1
LINGKUNGAN PEMROGRAMAN PYTHON

Bab ini membangun fondasi kerja yang dapat direproduksi: bagaimana interpreter, paket, lingkungan terisolasi, kontainer, pengujian, pengendalian versi, dan notebook ditempatkan dalam satu alur pengembangan perangkat lunak kimia kuantum. Fokusnya bukan menghafal perintah, melainkan memahami batas tanggung jawab setiap lapisan sehingga perhitungan dapat dijalankan ulang, diaudit, dan dipelihara oleh orang lain.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- menjelaskan hubungan runtime, paket, modul, dan antarmuka baris perintah
- membangun lingkungan terisolasi dengan venv atau Conda dan mendokumentasikan dependensi
- mengemas alur kerja ke dalam Docker tanpa menyembunyikan asumsi ilmiah
- menerapkan Git, pengujian, pemeriksaan statis, dan CI pada proyek komputasi
- menggunakan IPython dan Jupyter secara disiplin untuk eksplorasi yang tetap dapat direproduksi

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

1.1 Sistem Paket Python

Sistem paket menyatukan kode, metadata, dependensi, dan titik masuk. Dalam proyek ilmiah, paket bukan hanya cara distribusi, tetapi unit yang mendefinisikan ruang nama, versi, dan kontrak penggunaan.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

1.1.1 Runtime Python

Runtime adalah lapisan yang mengeksekusi bytecode, mengelola objek, memori, impor, dan pengecualian. Dalam komputasi kimia kuantum, pemahaman runtime membantu membedakan biaya interpretasi Python dari biaya kernel numerik BLAS, integral, atau FFT yang berjalan dalam kode terkompilasi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.1.2 Paket, Subpaket, dan Modul

Modul memisahkan tanggung jawab, paket membentuk ruang nama, sedangkan subpaket menata domain seperti integral, SCF, korelasi elektron, dan analisis sifat. Struktur impor yang jelas mengurangi ketergantungan melingkar dan memudahkan pengujian unit.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.1.3 Program Python sebagai Alat CLI

Antarmuka baris perintah membuat satu perhitungan dapat dipanggil dari terminal, skrip batch, penjadwal klaster, atau pipeline CI. Argumen harus memisahkan input ilmiah, pilihan numerik, lokasi keluaran, dan tingkat verbositas.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.1.4 Lingkungan Virtual

Lingkungan virtual mengisolasi dependensi proyek agar perubahan versi satu pustaka tidak merusak proyek lain. Berkas requirements atau lock file perlu diperlakukan sebagai bagian dari rekam jejak ilmiah.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat

hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.1.5 Conda

Conda mengelola paket Python dan pustaka biner non-Python, misalnya BLAS, LAPACK, compiler runtime, dan CUDA. Keunggulan ini penting pada perangkat lunak kimia kuantum yang menggabungkan banyak lapisan bahasa dan arsitektur.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.2 Program Python dalam Docker

Docker merekam filesystem dan dependensi biner dalam image. Ia meningkatkan portabilitas, tetapi tidak otomatis menjamin reproduksibilitas numerik karena hasil masih dipengaruhi arsitektur CPU/GPU, thread, dan pustaka matematika.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 1.2 program python dalam docker akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

1.3 Mengelola Proyek Python

Pengelolaan proyek bertujuan mengurangi entropi kode seiring bertambahnya fitur, kontributor, dan target platform. Proyek yang sehat memiliki sumber,

pengujian, dokumentasi, konfigurasi build, serta kebijakan rilis yang terpisah jelas.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

1.3.1 Pengendalian Versi

Git mencatat evolusi kode, konfigurasi, dan dokumentasi. Commit yang kecil dan bermakna memungkinkan hasil numerik ditelusuri kembali ke keadaan kode tertentu.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.3.2 Pengujian Program Python

Pengujian tidak hanya memeriksa bahwa program berjalan, tetapi juga bahwa energi, simetri, dimensi tensor, dan sifat invariansi memenuhi toleransi yang telah ditentukan.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.3.3 Gaya Kode dan Pemeriksaan Statis

Formatter, linter, dan type checker mengurangi variasi gaya serta mendeteksi kelas kesalahan sebelum runtime. Type hint sangat berguna untuk membedakan bentuk array, tipe indeks, dan objek konfigurasi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.

- Dokumentasikan asumsi dan batas penggunaan.

1.3.4 Merilis Paket Python

Rilis yang baik memiliki versi semantik, changelog, metadata, dokumentasi, dan artefak yang dapat dipasang. Pengguna harus dapat mengetahui perubahan API dan perubahan yang berpotensi memengaruhi hasil numerik.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.3.5 Alur Kerja CI dan DevOps

CI menjalankan pengujian, pemeriksaan gaya, dan pembangunan dokumentasi secara otomatis. Untuk proyek ilmiah, matriks pengujian sebaiknya mencakup sistem operasi, versi Python, backend numerik, dan pengujian hasil referensi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.4 Desain Modular dan Pemrograman Berorientasi Objek

Desain modular menempatkan perubahan pada batas yang sempit. OOP berguna bila objek domain memiliki keadaan dan protokol yang jelas; komposisi sering lebih aman daripada hierarki pewarisan yang dalam.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

1.4.1 Method Resolution Order pada Pewarisan Majemuk

MRO menentukan urutan pencarian metode ketika suatu kelas mewarisi lebih dari satu induk. Pemahaman ini mencegah perilaku tak terduga pada kelas metode elektronik yang menggabungkan fitur simetri, solvation, relativistik, dan korelasi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.4.2 Mixin

Mixin menambahkan perilaku sempit tanpa mewakili objek domain yang berdiri sendiri. Pola ini cocok untuk kemampuan tambahan seperti pencatatan waktu, penyimpanan checkpoint, atau evaluasi gradien.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.5 Bekerja dengan IPython dan Jupyter

Lingkungan interaktif mempercepat eksplorasi, tetapi keadaan tersembunyi dapat merusak reproducibility. Notebook perlu dijalankan dari awal secara berurutan dan dipisahkan dari kernel produksi.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

1.5.1 Konfigurasi Startup

Konfigurasi startup menyiapkan impor, formatter, presisi tampilan, dan kebijakan plotting secara konsisten. Isinya perlu minimal agar notebook tetap portabel.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.5.2 Ekstensi

Ekstensi memperluas fungsi IPython, misalnya autoreload dan profiler. Pengguna harus mencatat ekstensi yang memengaruhi keadaan sesi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.5.3 Magic Commands

Magic commands membantu pengukuran waktu, profiling, eksekusi shell, dan manajemen notebook. Hasilnya berguna untuk eksplorasi, tetapi eksperimen akhir sebaiknya dipindahkan ke skrip atau paket.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

1.5.4 Eksekusi Jarak Jauh

Notebook dapat menjadi antarmuka ke server atau klaster, tetapi kernel jarak jauh menuntut pengamanan koneksi, pemetaan berkas, dan pengelolaan sumber daya yang jelas.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Membuat paket qcflow dengan subperintah run, validate, dan report; menambahkan pyproject.toml, pengujian sederhana, serta image Docker yang menjalankan perhitungan H2.

Contoh kode Bab 1

```
# qcflow/cli.py
from __future__ import annotations
import argparse
from pathlib import Path

def build_parser() -> argparse.ArgumentParser:
    parser = argparse.ArgumentParser(prog="qcflow")
    parser.add_argument("input", type=Path, help="Berkas konfigurasi JSON")
    parser.add_argument("--workdir", type=Path, default=Path("run"))
    parser.add_argument("--dry-run", action="store_true")
    return parser

def main() -> int:
    args = build_parser().parse_args()
    args.workdir.mkdir(parents=True, exist_ok=True)
    print(f"Input: {args.input.resolve()}")
    print(f"Direktori kerja: {args.workdir.resolve()}")
    return 0

if __name__ == "__main__":
    raise SystemExit(main())
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Lingkungan eksekusi merupakan bagian dari metode ilmiah, bukan sekadar urusan instalasi.
- Pemisahan modul, pengujian, dan rekam versi mengurangi kesalahan yang sulit dilacak.
- Notebook efektif untuk eksplorasi, sedangkan paket dan CLI lebih tepat untuk produksi dan otomasi.

Latihan dan Refleksi

1. Jelaskan perbedaan tanggung jawab antara runtime Python, NumPy, dan pustaka BLAS ketika melakukan perkalian matriks.
2. Rancang struktur paket untuk metode SCF, MP2, dan analisis populasi tanpa ketergantungan melingkar.
3. Tuliskan strategi CI yang membandingkan energi referensi dengan toleransi absolut dan relatif.

Proyek mini: Membuat paket qcflow dengan subperintah run, validate, dan report; menambahkan pyproject.toml, pengujian sederhana, serta image Docker yang menjalankan perhitungan H2.

BAB 2
PEMROSESAN DATA

Data kimia kuantum hadir sebagai vektor orbital, matriks overlap, tensor integral, tabel hasil optimasi, dan kumpulan metadata. Bab ini membahas bagaimana NumPy dan Pandas memodelkan data tersebut, kapan operasi tervectorisasi menguntungkan, serta bagaimana bentuk, tipe data, view, copy, dan nilai hilang memengaruhi kebenaran maupun kinerja.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- menggunakan array NumPy secara vektoris dan memahami broadcasting
- memilih tipe data yang sesuai dengan kebutuhan presisi dan memori
- membedakan view dan copy sehingga mutasi tidak menghasilkan efek samping tersembunyi
- mengelola data berlabel dengan Series dan DataFrame
- membangun pipeline agregasi hasil komputasi yang dapat diaudit

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

2.1 Pemrosesan Data Tervektorisasi dengan NumPy

NumPy menyediakan model array homogen dan operator yang dekat dengan notasi matematis. Kinerja berasal dari kernel terkompilasi, operasi blok, dan akses memori teratur, bukan dari sintaks yang lebih singkat semata.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

2.1.1 Dasar-Dasar NumPy

ndarray menyimpan data homogen dalam blok memori dengan bentuk, stride, dan dtype. Operasi array memindahkan perulangan dari interpreter Python ke kernel terkompilasi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat

hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.1.2 Universal Functions (ufunc)

Ufunc menerapkan operasi elemen demi elemen dan mendukung broadcasting, casting, serta parameter out. Ufunc dapat dirangkai untuk membangun fungsi basis atau faktor redaman tanpa loop Python eksplisit.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.1.3 Operasi In-place

Operasi in-place mengurangi alokasi sementara, tetapi dapat merusak nilai yang masih diperlukan atau memicu casting yang tidak diinginkan. Pengguna perlu memahami kepemilikan data sebelum memodifikasi array.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.1.4 Broadcasting

Broadcasting menyelaraskan dimensi dari sisi kanan dan mereplikasi secara konseptual tanpa menyalin data. Teknik ini sangat efektif untuk jarak berpasangan dan evaluasi fungsi pada grid.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.1.5 Fancy Indexing

Fancy indexing memilih elemen menggunakan array indeks atau mask logis dan biasanya menghasilkan salinan. Biaya memori perlu diperhitungkan pada tensor integral berukuran besar.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.1.6 Masked Array

Masked array memisahkan nilai data dari penanda validitas. Dalam analisis hasil, mask dapat menandai geometri gagal konvergen tanpa mencampuradukkan kegagalan dengan nilai nol.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.1.7 Struktur Data ndarray NumPy

Bentuk, stride, order C/F, alignment, dan contiguity menentukan pola akses memori. Kernel linear algebra sering bekerja paling baik pada tata letak yang sesuai dengan antarmuka Fortran.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.1.8 View Array

View berbagi buffer dengan array asal. Transpose, slicing dasar, dan reshape tertentu dapat membentuk view sehingga perubahan pada salah satu objek terlihat pada objek lain.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.1.9 Fungsi reshape

reshape mengubah interpretasi dimensi tanpa selalu memindahkan data. Penggabungan indeks orbital perlu mempertahankan konvensi urutan agar kontraksi tensor tetap benar.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.2 Tipe Data dalam NumPy

Tipe data menentukan presisi, jangkauan, ukuran memori, dan aturan operasi. Dalam kimia kuantum, complex128, float64, dan integer indeks memiliki peran berbeda serta tidak boleh ditukar secara implisit.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

2.2.1 Type Casting

Casting menentukan aturan perubahan tipe, misalnya dari float64 ke float32 atau complex128. Casting diam-diam dapat menghilangkan bagian imajiner atau presisi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.2.2 Tipe Skalar dan Array Nol-Dimensi

Skalar NumPy dan array 0-D mirip tetapi tidak identik dalam indexing, serialisasi, dan dispatch fungsi. Perbedaan ini muncul pada hasil reduksi dan API yang mengharapkan ndarray.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.2.3 Infinity dan Not-a-Number

inf dan nan menyebar melalui banyak operasi. Deteksi eksplisit diperlukan agar divergensi SCF atau pembagian oleh eigenvalue kecil tidak dianggap sebagai hasil sah.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.2.4 Data Presisi Tinggi

Presisi tinggi dapat memakai longdouble, decimal, mpmath, atau aritmetika simbolik. Pemilihan harus didasarkan pada sumber galat, bukan asumsi bahwa lebih banyak digit selalu menyelesaikan masalah.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.2.5 Structured Array

Structured array menyimpan record heterogen dalam satu ndarray. Format ini berguna untuk interoperabilitas biner, tetapi DataFrame sering lebih nyaman untuk analisis berlabel.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.3 Data Berlabel: Pandas

Pandas menambah semantik label pada data tabel. Ia sangat cocok untuk eksperimen komputasi, tetapi bukan tempat ideal untuk kontraksi tensor intensif.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

2.3.1 Objek Data Pandas

Series dan DataFrame menambahkan indeks serta nama kolom pada array. Label membantu menghubungkan energi, metode, basis, muatan, spin, dan status konvergensi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.3.2 Broadcasting pada Pandas

Operasi Pandas menyelaraskan label sebelum menghitung. Keselarasan otomatis kuat, tetapi dapat menciptakan NaN jika indeks tidak cocok.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.3.3 Indexing

loc bekerja berbasis label, iloc berbasis posisi. Pemisahan ini mencegah ambigu ketika indeks berupa angka yang bukan nomor baris.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.3.4 Metode query dan eval

query dan eval menyatakan penyaringan dan ekspresi kolom secara ringkas. Ekspresi tetap perlu divalidasi apabila berasal dari input pengguna.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.3.5 Mengubah Struktur DataFrame

pivot, melt, stack, dan unstack memindahkan data antara format lebar dan panjang. Format panjang biasanya lebih sesuai untuk visualisasi dan model statistik.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.3.6 Tipe Data

Pandas memiliki dtype nullable, kategorikal, datetime, dan string. Kategori menghemat memori untuk label metode atau basis yang berulang.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat

hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.3.7 Data Hilang

Nilai hilang dapat berarti perhitungan gagal, tidak dijalankan, atau tidak relevan. Makna tersebut sebaiknya disimpan dalam kolom status, bukan dibiarkan hanya sebagai NaN.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

2.3.8 Pengelompokan dan Agregasi

groupby merangkum hasil berdasarkan metode, basis, molekul, atau konfigurasi. Agregasi harus menyertakan ukuran sampel dan ukuran penyebaran, bukan hanya rata-rata.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

2.3.9 View dan Copy

Pandas dapat mengembalikan view atau copy bergantung pada operasi. Penggunaan assignment eksplisit dengan loc mengurangi risiko chained assignment.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Menganalisis pemindaian panjang ikatan: menyimpan energi berbagai metode dalam DataFrame, menandai kegagalan, menghitung energi relatif, dan menemukan minimum untuk setiap metode.

Contoh kode Bab 2

```
import numpy as np
import pandas as pd

r = np.linspace(0.5, 2.5, 81)
energy = -1.0 + 0.7 * (r - 0.74)**2
energy[70:] = np.nan  # contoh titik yang gagal

df = pd.DataFrame({"r_angstrom": r, "energy_hartree": energy})
df["status"] = np.where(df["energy_hartree"].isna(), "gagal", "ok")
valid = df.query("status == 'ok'").copy()
valid["delta_E_kJmol"] = (valid["energy_hartree"] -
valid["energy_hartree"].min()) * 2625.4996
minimum = valid.loc[valid["energy_hartree"].idxmin()]
print(minimum)
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Vektorisasi memindahkan kerja ke kernel terkompilasi, tetapi bentuk dan alokasi tetap harus dipahami.
- Dtype, view, copy, serta nilai nonhingga dapat mengubah hasil secara halus.
- Pandas tepat untuk metadata dan analisis berlabel; tensor numerik tetap lebih efisien dalam NumPy.

Latihan dan Refleksi

4. Tunjukkan dengan contoh kapan slicing menghasilkan view dan fancy indexing menghasilkan copy.
5. Rancang skema DataFrame untuk menyimpan hasil optimasi geometri dari beberapa molekul dan metode.
6. Bandingkan penggunaan float32 dan float64 untuk diagonalization matriks simetris yang hampir degenerat.

Proyek mini: Menganalisis pemindaian panjang ikatan: menyimpan energi berbagai metode dalam DataFrame, menandai kegagalan, menghitung energi relatif, dan menemukan minimum untuk setiap metode.

BAB 3
VISUALISASI

Visualisasi menjembatani angka dengan struktur fisik: kurva energi, spektrum, permukaan, orbital, densitas, dan grid volumetrik. Bab ini menekankan pemilihan representasi, pemisahan data dari tampilan, dan ekspor ke format yang dapat dibaca oleh perangkat lunak kimia.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- membuat visualisasi dua dimensi yang informatif dengan Matplotlib
- memanfaatkan plotting Pandas untuk eksplorasi cepat
- membuat representasi tiga dimensi dengan Mayavi atau alternatifnya
- menulis data dalam format Molden dan cube
- menggunakan template untuk menghasilkan laporan visual yang konsisten

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

3.1 Matplotlib

Matplotlib memisahkan Figure, Axes, dan artist. Pola object-oriented membuat kontrol layout lebih stabil daripada mengandalkan state global pyplot dalam proyek besar.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 3.1 matplotlib akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

3.2 Visualisasi Pandas

Plotting Pandas cocok untuk inspeksi awal karena kolom dan indeks langsung menjadi seri visual. Untuk publikasi, objek Axes yang dihasilkan sebaiknya disempurnakan melalui Matplotlib.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 3.2 visualisasi pandas akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

3.3 Mayavi untuk Plot 3D

Mayavi menggunakan pipeline VTK untuk isosurface, volume, dan glyph tiga dimensi. Data volumetrik perlu dipetakan ke koordinat fisik, bukan sekadar indeks grid.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 3.3 mayavi untuk plot 3d akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

3.4 Visualisasi Kimia Kuantum

Visualisasi orbital dan densitas memerlukan data geometri, basis atau grid, serta konvensi tanda dan satuan. Representasi harus dibedakan antara kuantitas observabel dan fungsi gelombang yang tandanya arbitrer.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

3.4.1 Template Jinja

Jinja memisahkan data hasil dari struktur laporan. Template HTML dapat menghasilkan halaman ringkas yang konsisten untuk banyak molekul tanpa merangkai string secara manual.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

3.4.2 Format Molden

Format Molden menyimpan atom, basis, orbital, energi, dan koefisien dalam representasi teks yang dikenal banyak visualizer. Implementasi perlu mengikuti urutan basis dan konvensi normalisasi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

3.4.3 Format Cube

Cube menyimpan grid tiga dimensi bersama metadata geometri. Ukuran berkas tumbuh sebagai hasil kali jumlah titik pada ketiga sumbu sehingga resolusi perlu dipilih secara rasional.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat

hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Membuat kurva energi potensial, peta kontur orbital model, dan penulis cube sederhana untuk data skalar pada grid Kartesian.

Contoh kode Bab 3

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-4.0, 4.0, 240)
y = np.linspace(-4.0, 4.0, 240)
X, Y = np.meshgrid(x, y, indexing="xy")
psi = X * np.exp(-0.7 * (X**2 + Y**2))

fig, ax = plt.subplots(figsize=(6, 5))
contour = ax.contourf(X, Y, psi, levels=31)
ax.set(xlabel="x / bohr", ylabel="y / bohr", title="Model irisan orbital p")
fig.colorbar(contour, ax=ax, label="amplitudo")
fig.tight_layout()
fig.savefig("orbital_model.png", dpi=200)
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Visualisasi harus mempertahankan satuan, skala, dan konteks fisik.
- Format kimia seperti Molden dan cube menuntut konsistensi urutan basis dan grid.
- Template memisahkan presentasi dari perhitungan sehingga pelaporan lebih mudah dipelihara.

Latihan dan Refleksi

7. Buat grafik energi relatif dengan garis referensi pada minimum dan label satuan yang lengkap.
8. Jelaskan perbedaan antara isosurface densitas dan isosurface orbital.
9. Hitung kebutuhan memori cube berukuran 200×200×200 dengan float64.

Proyek mini: Membuat kurva energi potensial, peta kontur orbital model, dan penulis cube sederhana untuk data skalar pada grid Kartesian.

BAB 4
PERANGKAT KOMPUTASI ILMIAH

Perangkat komputasi ilmiah menyediakan blok bangunan bagi algoritma kimia kuantum: aljabar linear, matriks jarang, operator linear, kontraksi tensor, dan transformasi Fourier. Bab ini mengaitkan API dengan struktur matematis dan pola biaya komputasi.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- memilih dekomposisi atau solver sesuai struktur masalah
- mengenali kapan matriks jarang dan LinearOperator menghemat memori
- menuliskan kontraksi tensor secara jelas dan efisien
- menggunakan FFT dengan normalisasi dan tata letak yang benar
- melakukan benchmark tanpa mencampur biaya setup dan eksekusi

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

4.1 Aljabar Linear

Masalah struktur elektronik didominasi eigenproblem, faktorisasi, least squares, dan solver linear. Memanfaatkan simetri Hermitian atau positive-definite menghasilkan algoritma yang lebih stabil dan efisien.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 4.1 aljabar linear akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

4.2 Matriks Jarang

Matriks jarang hanya menguntungkan bila pola nonnol cukup sedikit dan operasi memanfaatkannya. Overhead indeks dapat membuat representasi jarang lebih buruk untuk matriks kecil atau cukup padat.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

4.2.1 Format Penyimpanan

CSR efisien untuk operasi baris dan matvec, CSC untuk operasi kolom, sedangkan COO nyaman untuk konstruksi. Pemilihan format memengaruhi biaya konversi dan locality.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

4.2.2 Aljabar Linear untuk Matriks Jarang

Solver iteratif mengeksploitasi matvec dan preconditioner tanpa membentuk faktorisasi padat. Konvergensi bergantung pada spektrum dan kualitas preconditioner.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

4.2.3 Operator Linear

LinearOperator mewakili aksi matriks melalui fungsi matvec. Pendekatan matrix-free penting ketika Hamiltonian terlalu besar untuk disimpan.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

4.3 Kontraksi Tensor

Kontraksi tensor menggeneralisasi perkalian matriks. Penentuan jalur kontraksi mengubah kebutuhan memori dan jumlah operasi dengan orde yang besar.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 4.3 kontraksi tensor akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

4.4 Transformasi Fourier Diskret

DFT menghubungkan representasi ruang nyata dan ruang frekuensi. FFT menurunkan kompleksitas, tetapi grid, normalisasi, periodisitas, dan ordering frekuensi harus konsisten.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

4.4.1 FFT SciPy

scipy.fft menyediakan transformasi multidimensi, backend, dan pilihan worker. Normalisasi harus konsisten antara transformasi maju dan balik.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

4.4.2 FFT Intel

Implementasi vendor dapat memanfaatkan instruksi dan threading khusus prosesor. Keuntungan aktual bergantung pada ukuran grid, layout, dan biaya thread.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

4.4.3 PyFFTW

PyFFTW memberi akses ke FFTW plan dan wisdom. Waktu perencanaan dapat diamortisasi jika bentuk transformasi digunakan berulang kali.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

4.4.4 Benchmark Kinerja FFT

Benchmark harus melakukan warm-up, mengulang cukup banyak kali, memisahkan alokasi, dan memeriksa kebenaran numerik.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Lakukan warm-up dan laporkan median serta variasi waktu.
- Pisahkan waktu setup, alokasi, kompilasi, dan kernel.

4.4.5 Memilih Pustaka FFT

Pilihan backend mempertimbangkan portabilitas, lisensi, ukuran transformasi, threading, GPU, serta biaya integrasi, bukan hanya angka tercepat pada satu mesin.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat

hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Menyelesaikan persamaan eigen umum FC = SCE, membangun operator matrix-free, dan membandingkan FFT pada beberapa ukuran grid.

Contoh kode Bab 4

```
import numpy as np
from scipy import linalg

# Contoh masalah eigen umum yang muncul dalam Roothaan-Hall
F = np.array([[-0.9, -0.1], [-0.1, -0.4]])
S = np.array([[1.0, 0.2], [0.2, 1.0]])

eps, C = linalg.eigh(F, S)
residual = np.linalg.norm(F @ C - S @ C @ np.diag(eps))
print("energi orbital:", eps)
print("norma residual:", residual)
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Struktur matematis menentukan algoritma yang tepat.
- Representasi matrix-free menukar penyimpanan dengan evaluasi operator.
- Kontraksi tensor dan FFT perlu dianalisis dari aspek bentuk, layout, dan alokasi sementara.

Latihan dan Refleksi

10. Jelaskan mengapa eigensolver simetris lebih tepat daripada solver umum untuk matriks Fock ortogonal.
11. Buat LinearOperator untuk matriks diagonal ditambah rank-one update.
12. Bandingkan kompleksitas kontraksi ijab,ab->ij dengan pembentukan tensor perantara.

Proyek mini: Menyelesaikan persamaan eigen umum FC = SCE, membangun operator matrix-free, dan membandingkan FFT pada beberapa ukuran grid.

BAB 5
METAPROGRAMMING DAN KOMPUTASI NON-NUMERIK

Tidak semua persoalan ilmiah diselesaikan dengan operasi numerik langsung. Pembuatan kode, manipulasi AST, aljabar simbolik, dan diferensiasi otomatis dapat menghasilkan program, persamaan, atau turunan baru dari representasi yang lebih abstrak.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- membedakan evaluasi kode dinamis dari pembuatan kode yang terkontrol
- memanipulasi fungsi dan kelas pada runtime secara bertanggung jawab
- menggunakan AST untuk inspeksi dan transformasi kode
- menerapkan komputasi simbolik untuk menyederhanakan ekspresi
- memahami mode diferensiasi otomatis dan implikasinya bagi memori

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

5.1 Pembuatan Kode

Pembuatan kode menukar fleksibilitas runtime dengan kernel yang lebih khusus. Keuntungan diperoleh ketika struktur masalah diketahui lebih awal dan dipakai berulang kali.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

5.1.1 eval dan exec

eval mengevaluasi ekspresi, sedangkan exec menjalankan blok pernyataan. Keduanya berisiko bila menerima input tak tepercaya dan sering dapat diganti dengan mapping fungsi atau parser khusus.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

5.1.2 Menyusun Kode Secara Programatis

Kode dapat dibangun dari template, node AST, atau intermediate representation. Tujuannya biasanya menghilangkan cabang, mengkhususkan dimensi, atau menghasilkan kontraksi dari persamaan simbolik.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

5.1.3 Memanipulasi Kelas dan Fungsi saat Runtime

Decorator, descriptor, monkey patch, dan pembuatan kelas dinamis memungkinkan komposisi perilaku. Fleksibilitas ini harus diimbangi dokumentasi serta pengujian karena alur resolusi menjadi kurang eksplisit.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

5.1.4 AST Python

AST merepresentasikan struktur sintaks sebelum kompilasi bytecode. Transformasi AST dapat menambahkan instrumentation atau mengganti operasi, tetapi harus mempertahankan lokasi, scope, dan semantik.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

5.2 Komputasi Simbolik

Sistem simbolik mempertahankan struktur ekspresi, memungkinkan diferensiasi, penyederhanaan, substitusi, dan code generation. Ledakan ukuran ekspresi harus dikendalikan dengan common-subexpression elimination.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 5.2 komputasi simbolik akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

5.3 Diferensiasi Otomatis

AD menerapkan aturan rantai pada program. Ia berbeda dari finite difference karena menghasilkan turunan sampai presisi mesin untuk operasi yang terdiferensiasi.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

5.3.1 PyTorch

PyTorch merekam graf operasi dinamis dan menghitung gradien reverse-mode. Ia cocok untuk keluaran skalar dengan banyak parameter, tetapi mutasi in-place tertentu dapat mengganggu graf.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

5.3.2 JAX

JAX menggabungkan transformasi fungsi seperti grad, jit, vmap, dan pmap. Kode perlu bersifat fungsional, bentuk array stabil, dan efek samping diminimalkan agar tracing dapat diprediksi.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Menyusun ekspresi energi model dengan SymPy, menurunkannya secara simbolik, lalu membandingkan turunan tersebut dengan gradien JAX.

Contoh kode Bab 5

```
import sympy as sp

x, a, b = sp.symbols("x a b", positive=True)
energy = a * sp.exp(-b*x**2) + x**4
gradient = sp.diff(energy, x)
optimized = sp.cse([energy, gradient])
print("dE/dx =", gradient)
print("common subexpressions =", optimized[0])
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Metaprogramming memindahkan sebagian pekerjaan dari waktu eksekusi ke tahap pembentukan program.
- Komputasi simbolik mempertahankan struktur aljabar yang hilang pada array numerik.
- Diferensiasi otomatis menghitung turunan program dan menuntut disiplin pada aliran data serta efek samping.

Latihan dan Refleksi

13. Jelaskan mengapa exec tidak tepat untuk membaca file konfigurasi.
14. Rancang intermediate representation sederhana untuk kontraksi tensor.

15. Bandingkan forward-mode dan reverse-mode untuk fungsi dengan 1000 input dan satu keluaran.

Proyek mini: Menyusun ekspresi energi model dengan SymPy, menurunkannya secara simbolik, lalu membandingkan turunan tersebut dengan gradien JAX.

BAB 6
MASUKAN DAN KELUARAN

Perhitungan ilmiah jarang berdiri sendiri. Data harus diserialkan, disimpan, dibaca ulang, dipertukarkan melalui jaringan, dan dilindungi dari korupsi atau ketidakcocokan versi. Bab ini menyusun strategi I/O berdasarkan ukuran, struktur, portabilitas, dan pola akses.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- memilih format serialisasi berdasarkan keamanan dan interoperabilitas
- menggunakan npy, HDF5, dan memory mapping secara tepat
- memahami buffering serta I/O dalam memori
- merancang API jaringan yang memisahkan identitas pekerjaan dari data besar
- menerapkan checkpoint yang dapat diverifikasi

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

6.1 Serialisasi

Serialisasi mengubah objek menjadi representasi yang dapat disimpan atau dikirim. Desain skema dan versi menentukan apakah data lama tetap dapat dibaca.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

6.1.1 Pickle

Pickle mempertahankan banyak objek Python, tetapi tidak aman untuk data tak tepercaya dan rapuh terhadap perubahan kelas. Format ini cocok untuk cache internal yang terkendali, bukan pertukaran publik.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.

- Dokumentasikan asumsi dan batas penggunaan.

6.1.2 JSON

JSON bersifat tekstual dan lintas bahasa, ideal untuk metadata serta konfigurasi. Array besar sebaiknya disimpan terpisah karena JSON tidak efisien untuk data biner dan tidak memiliki tipe kompleks bawaan.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.1.3 YAML

YAML mudah dibaca dan mendukung struktur kaya, tetapi parser harus menggunakan mode aman. Kompleksitas sintaks dapat menimbulkan interpretasi tipe yang mengejutkan.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.2 I/O Berkas

Pemilihan format berkas berkaitan dengan ukuran, struktur, akses parsial, dan interoperabilitas. Checksum dan penulisan atomik meningkatkan integritas.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

6.2.1 Format npy

npy menyimpan dtype, bentuk, dan urutan array secara langsung. Format ini cepat untuk pertukaran NumPy dan lebih andal daripada dump teks untuk tensor besar.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat

hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.2.2 Penyimpanan HDF5

HDF5 mengatur dataset dalam hierarki, mendukung chunking, kompresi, atribut, dan pembacaan parsial. Desain chunk harus mengikuti pola akses dominan.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.2.3 Memory Mapping

Memory mapping mengakses bagian berkas seolah-olah array memori. Teknik ini mengurangi kebutuhan RAM, tetapi pola akses acak dapat memicu page fault yang mahal.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.3 Buffering I/O

Buffering menggabungkan operasi kecil menjadi transfer lebih besar. Flush yang terlalu sering menurunkan throughput, sedangkan buffer yang tidak pernah di-flush berisiko kehilangan data.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.

- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 6.3 buffering i/o akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

6.4 I/O dalam Memori

BytesIO dan StringIO memungkinkan API berkas bekerja tanpa disk. Teknik ini berguna untuk pengujian, kompresi, dan pengiriman data melalui jaringan.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 6.4 i/o dalam memori akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

6.5 I/O Jaringan

Jaringan menambahkan latency, kegagalan parsial, dan perbedaan versi. Protokol harus dirancang untuk retry yang aman dan autentikasi.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

6.5.1 Permintaan Jaringan melalui HTTP

HTTP cocok untuk transfer sumber daya dan API. Timeout, retry, checksum, dan idempotensi harus ditentukan eksplisit.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat

hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.5.2 REST API

REST memodelkan sumber daya melalui URL dan metode HTTP. Pekerjaan komputasi panjang biasanya memerlukan pola submit-status-result, bukan koneksi tunggal yang menunggu.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.5.3 RPC melalui HTTP

RPC mengekspresikan pemanggilan prosedur jarak jauh secara langsung. Kontrak tipe dan versi API menjadi pusat interoperabilitas.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.5.4 gRPC

gRPC menggunakan Protocol Buffers dan HTTP/2 untuk kontrak terstruktur serta streaming. Ia kuat untuk layanan internal, tetapi menambah langkah generasi kode.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

6.5.5 Apache Arrow

Arrow mendefinisikan layout kolumnar lintas bahasa dan memungkinkan pertukaran data mendekati zero-copy. Ia cocok untuk tabel numerik, bukan pengganti umum untuk semua objek domain.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Membangun checkpoint perhitungan yang memisahkan metadata JSON dan tensor HDF5, dilengkapi hash SHA-256 serta nomor versi skema.

Contoh kode Bab 6

```
from __future__ import annotations
import hashlib
import json
from pathlib import Path
import h5py
import numpy as np

root = Path("checkpoint")
root.mkdir(exist_ok=True)
C = np.eye(4)
with h5py.File(root / "arrays.h5", "w") as h5:
    h5.create_dataset("mo_coeff", data=C, compression="gzip")

digest = hashlib.sha256((root / "arrays.h5").read_bytes()).hexdigest()
metadata = {"schema": 1, "method": "RHF", "basis": "sto-3g", "sha256": digest}
(root / "metadata.json").write_text(json.dumps(metadata, indent=2),
encoding="utf-8")
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Format data harus dipilih berdasarkan keamanan, portabilitas, ukuran, dan pola akses.
- Metadata dan tensor besar sering lebih baik disimpan dalam lapisan berbeda.
- I/O jaringan memerlukan kontrak, timeout, retry, dan verifikasi integritas.

Latihan dan Refleksi

16. Bandingkan pickle, JSON, npy, dan HDF5 untuk menyimpan objek hasil SCF.
17. Rancang chunk HDF5 untuk tensor dengan akses dominan per blok orbital virtual.
18. Jelaskan pola API submit-status-result untuk komputasi yang berlangsung satu jam.

Proyek mini: Membangun checkpoint perhitungan yang memisahkan metadata JSON dan tensor HDF5, dilengkapi hash SHA-256 serta nomor versi skema.

BAB 7
BEKERJA DENGAN CLOUD

Cloud menawarkan sumber daya elastis, penyimpanan objek, layanan antrean, dan eksekusi terdistribusi. Bab ini menempatkan cloud dalam konteks ilmiah: biaya transfer, lokalisasi data, reproducibility image, fault tolerance, dan pemisahan orkestrasi dari kernel kimia kuantum.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- membandingkan cloud dengan superkomputer berdasarkan karakter beban
- merancang workflow yang menyimpan keadaan dan dapat dilanjutkan
- memilih komputasi, penyimpanan, antrean, dan jaringan secara terpadu
- memahami Function-as-a-Service dan batasannya
- menggunakan Celery, Dask, atau Ray sesuai pola tugas

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

7.1 Memanfaatkan Komputasi Cloud

Cloud memisahkan sumber daya menjadi layanan yang dapat diprovisi melalui API. Keuntungan utamanya adalah elastisitas dan otomasi, bukan asumsi bahwa semua pekerjaan akan lebih cepat.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

7.1.1 Perbandingan Cloud dan Superkomputer

Cloud unggul pada elastisitas dan layanan terkelola, sedangkan superkomputer sering unggul pada interconnect, filesystem paralel, dan pekerjaan MPI skala besar. Pilihan ditentukan oleh pola komunikasi serta biaya total.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

7.1.2 Merancang Workflow

Workflow memecah penelitian menjadi tugas dengan input, output, dependensi, retry, dan status. Setiap tugas sebaiknya idempotent agar pengulangan tidak merusak data.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

7.1.3 Sumber Daya Komputasi

CPU, memori, GPU, disk lokal, dan jaringan harus dipilih sebagai satu paket. Mesin termurah per jam belum tentu termurah per hasil.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

7.1.4 Komunikasi Antarlayanan Cloud

Layanan bertukar pesan, objek, atau panggilan API. Pesan kecil sebaiknya membawa referensi ke data besar, bukan tensor besar itu sendiri.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

7.1.5 Function-as-a-Service

FaaS cocok untuk fungsi singkat dan stateless. Batas waktu, cold start, ukuran deployment, serta tidak adanya interconnect cepat membatasi penggunaannya untuk kernel berat.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

7.2 Eksekutor Pekerjaan Terdistribusi

Eksekutor mendistribusikan tugas dan menangani antrean, retry, serta pengumpulan hasil. Modelnya perlu disesuaikan dengan ukuran data dan dependensi.

Dari sudut rekayasa perangkat lunak, keputusan pada bagian ini harus dinyatakan melalui API, konfigurasi, dan pengujian yang jelas. Contoh kecil perlu dipisahkan dari asumsi lingkungan sehingga dapat dipindahkan dari notebook ke skrip, layanan, atau pipeline otomatis.

7.2.1 Celery

Celery menggunakan broker dan worker untuk antrean tugas. Ia tepat untuk orkestrasi pekerjaan independen dan retry, tetapi bukan pengganti MPI untuk komunikasi rapat.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

7.2.2 Dask

Dask membangun graf tugas dinamis serta array/dataframe terpartisi. Ia efektif untuk analisis dan komputasi berbasis blok dengan dependensi eksplisit.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

7.2.3 Ray

Ray menyediakan task, actor, dan object store terdistribusi. Actor berguna ketika worker perlu mempertahankan keadaan, misalnya cache integral atau model surrogate.

Praktik yang baik adalah membuat perilaku ini eksplisit melalui nama fungsi, parameter, metadata, dan dokumentasi. Hindari keadaan global yang membuat hasil bergantung pada urutan eksekusi. Ketika data besar terlibat, perhatikan apakah operasi membuat view, copy, buffer, atau koneksi baru.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Membuat pemindaian geometri terdistribusi berbasis Dask yang mengembalikan tabel hasil dan mencatat kegagalan setiap titik.

Contoh kode Bab 7

```
from dask import delayed, compute

@delayed
def energy_at(distance: float) -> dict[str, float]:
    # Ganti dengan pemanggilan kernel kimia kuantum yang sebenarnya.
    energy = -1.0 + 0.65 * (distance - 0.74)**2
    return {"r": distance, "energy": energy}

jobs = [energy_at(0.5 + 0.05*i) for i in range(40)]
results = compute(*jobs, scheduler="threads")
print(min(results, key=lambda row: row["energy"]))
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Cloud dan HPC tradisional memiliki kekuatan berbeda; pola komunikasi adalah pembeda penting.
- Workflow yang idempotent dan berstatus eksplisit lebih tahan terhadap kegagalan.
- Celery, Dask, dan Ray menyasar pola distribusi yang berbeda.

Latihan dan Refleksi

19. Bandingkan biaya transfer data dengan biaya komputasi untuk tensor integral berukuran 100 GB.
20. Rancang workflow optimasi geometri yang dapat dilanjutkan setelah worker gagal.
21. Pilih antara Celery, Dask, Ray, dan MPI untuk empat skenario yang berbeda.

Proyek mini: Membuat pemindaian geometri terdistribusi berbasis Dask yang mengembalikan tabel hasil dan mencatat kegagalan setiap titik.

BAGIAN 2

KOMPUTASI KINERJA TINGGI DENGAN PYTHON

Bagian ini membahas integrasi bahasa, profiling, optimasi, paralelisme CPU, dan GPU dengan fokus pada model biaya dan pemindahan data.

BAB 8
ANTARMUKA BAHASA ASING

Kinerja dan interoperabilitas sering menuntut Python berkomunikasi dengan C, C++, Fortran, Rust, atau proses lain. Bab ini membahas batas ABI, representasi tipe, kepemilikan memori, dan kesalahan yang muncul ketika dua runtime berbagi data.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- menentukan kapan IPC lebih tepat daripada pemanggilan fungsi langsung
- memetakan tipe skalar dan array antara Python dan C/C++
- membandingkan Cython, pybind11, ctypes, dan CFFI
- memanggil Fortran dengan aman dan memahami urutan memori
- mengenali kebocoran memori serta konflik simbol

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

8.1 Antarmuka Komunikasi Antarproses

IPC memisahkan crash dan runtime antarbahasa dengan biaya serialisasi atau shared memory. Ia sering lebih mudah dipelihara daripada binding langsung untuk program besar.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 8.1 antarmuka komunikasi antarproses akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

8.2 Antarmuka C/C++

Binding C/C++ memberi akses ke kernel terkompilasi dan pustaka lama. Tantangan utama berada pada ABI, lifetime objek, GIL, dan distribusi binary wheel.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

8.2.1 Tipe Data dan Konversi Tipe

Batas bahasa harus menentukan ukuran integer, presisi floating-point, alignment, signedness, dan kepemilikan. Buffer protocol dan memoryview dapat menghindari salinan jika layout kompatibel.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

8.2.2 Cython

Cython menambahkan deklarasi tipe dan menerjemahkan kode ke C. Keuntungan terbesar muncul ketika loop dan indexing dapat dikompilasi tanpa interaksi objek Python.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

8.2.3 pybind11

pybind11 membungkus C++ modern dengan template dan konversi tipe yang nyaman. Binding tetap perlu menetapkan lifetime dan pelepasan GIL untuk fungsi panjang.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai

yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

8.2.4 Mengompilasi Kode C++ menjadi Modul Python

Sistem build harus mengelola compiler, ABI, header NumPy, dan wheel lintas platform. Build isolation mengurangi ketergantungan pada lingkungan lokal.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

8.3 Foreign Function Interface

FFI memanggil fungsi berdasarkan deklarasi ABI tanpa wrapper lengkap. Ia cepat dikembangkan, tetapi type mismatch dapat menghasilkan crash yang sulit didiagnosis.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

8.3.1 ctypes

ctypes memanggil pustaka dinamis tanpa kompilasi binding khusus. argtypes dan restype harus ditetapkan agar pointer tidak ditafsirkan keliru.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

8.3.2 CFFI

CFFI menyediakan mode ABI dan API. Mode API dikompilasi dan biasanya memberi pemeriksaan lebih baik atas deklarasi C.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

8.3.3 Kebocoran Memori

Kebocoran terjadi ketika alokasi tidak dilepas atau referensi lintas bahasa mempertahankan objek. Pengujian berulang dan alat profiler memori diperlukan.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

8.3.4 Nama Fungsi Duplikat dalam Ekstensi

Simbol global yang sama dapat berbenturan ketika beberapa ekstensi memuat pustaka berbeda. Visibility, namespacing, dan linking yang disiplin mengurangi risiko.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

8.4 Antarmuka Fortran

Fortran tetap penting dalam kimia kuantum karena ekosistem numerik dan kode legacy. Urutan kolom serta compiler ABI harus dipahami.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

8.4.1 ctypes untuk Fortran

Nama simbol Fortran, pass-by-reference, dan tata letak kolom perlu diperhatikan. Wrapper C sering menyederhanakan ABI lintas compiler.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

8.4.2 Kompiler f2py

f2py menghasilkan wrapper NumPy untuk subrutin Fortran. Intent dan dimensi yang jelas membantu f2py membuat antarmuka yang aman.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

8.5 Antarmuka Rust

Rust menawarkan keamanan memori statis dan abstraksi zero-cost. Binding melalui PyO3 atau C ABI dapat menghasilkan ekstensi yang aman, meskipun toolchain dan ekosistemnya berbeda.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 8.5 antarmuka rust akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

Praktikum Terpandu

Membungkus fungsi C sederhana untuk menghitung dot product dan membandingkan salinan data dengan penggunaan pointer langsung.

Contoh kode Bab 8

```
# Contoh sisi Python untuk pustaka C yang telah dikompilasi
import ctypes as ct
import numpy as np
from numpy.ctypeslib import ndpointer

lib = ct.CDLL("./libdot.so")
lib.dot64.argtypes = [
    ndpointer(dtype=np.float64, flags="C_CONTIGUOUS"),
    ndpointer(dtype=np.float64, flags="C_CONTIGUOUS"),
    ct.c_size_t,
]
lib.dot64.restype = ct.c_double

x = np.arange(10.0, dtype=np.float64)
y = x[::-1].copy()  # pastikan kontigu
print(lib.dot64(x, y, x.size))
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Antarmuka lintas bahasa adalah kontrak tipe, layout, ABI, dan kepemilikan.
- Zero-copy hanya aman bila tipe, alignment, dan lifetime benar-benar kompatibel.
- Kemudahan binding harus diseimbangkan dengan portabilitas dan kemampuan debugging.

Latihan dan Refleksi

22. Jelaskan risiko memanggil fungsi C tanpa menetapkan restype.
23. Bandingkan Cython, pybind11, ctypes, dan CFFI untuk pustaka yang sudah tersedia.
24. Rancang pengujian yang mendeteksi kebocoran memori pada binding berulang.

Proyek mini: Membungkus fungsi C sederhana untuk menghitung dot product dan membandingkan salinan data dengan penggunaan pointer langsung.

BAB 9
OPTIMASI KINERJA PROGRAM

Optimasi yang benar dimulai dari model biaya, pengukuran, dan identifikasi bottleneck. Bab ini membedakan latensi dari throughput, compute-bound dari I/O-bound, dan perbaikan algoritmik dari mikro-optimasi. Teknik kompilasi, layout data, cache, memoization, serta lazy evaluation ditempatkan dalam urutan keputusan yang rasional.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- membuat model biaya sebelum mengoptimalkan
- menggunakan profiler yang sesuai dengan jenis bottleneck
- menulis Python yang bersahabat dengan cache dan kernel vektor
- membandingkan Numba, Cython, dan Pythran
- mengurangi biaya I/O, alokasi, dan perhitungan berulang

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

9.1 Prinsip Optimasi Kinerja

Optimasi harus diarahkan oleh bottleneck aktual. Perubahan yang mempercepat satu mikrobenchmark dapat memperlambat workload nyata atau mengurangi stabilitas numerik.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

9.1.1 Perbandingan Biaya Operasi Python dan C/C++

Operasi skalar Python membawa biaya dispatch dan objek yang jauh lebih besar daripada instruksi mesin. Namun satu operasi NumPy dapat mewakili jutaan operasi C sehingga granularitas menjadi kunci.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.1.2 Overhead Perangkat Keras dan Sistem Operasi

Cache, NUMA, page fault, context switch, frequency scaling, dan scheduler memengaruhi waktu. Benchmark perlu mengendalikan faktor tersebut sejauh mungkin.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.1.3 Latensi dan Throughput

Latensi mengukur waktu satu tugas, throughput mengukur jumlah tugas per satuan waktu. Batch dapat meningkatkan throughput dengan mengorbankan latensi.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.1.4 Strategi Mengoptimalkan Latensi dan Throughput

Untuk latensi, kurangi jalur kritis dan startup; untuk throughput, tingkatkan batching, paralelisme, dan pemanfaatan sumber daya.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.1.5 Compute-bound dan I/O-bound

Compute-bound dibatasi unit aritmetika, sedangkan I/O-bound menunggu data. Optimasi harus menargetkan sumber tunggu yang dominan.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.1.6 Instruction-Level Parallelism

CPU dapat mengeksekusi beberapa instruksi independen secara tumpang tindih. Dependensi data dan cabang mengurangi peluang ILP.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.2 Profiling

Profiler menjawab di mana waktu dan sumber daya digunakan. Tidak ada satu profiler untuk semua masalah; fungsi Python, native kernel, memori, dan hardware counter memerlukan alat berbeda.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

9.2.1 Uji Benchmark

Benchmark harus mengukur fungsi representatif, melakukan warm-up, mengulang, dan melaporkan distribusi waktu.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Lakukan warm-up dan laporkan median serta variasi waktu.
- Pisahkan waktu setup, alokasi, kompilasi, dan kernel.

9.2.2 Alat Profiling Bawaan Python

cProfile mengukur waktu per fungsi dan cocok untuk gambaran awal. Overheadnya perlu dipahami.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Lakukan warm-up dan laporkan median serta variasi waktu.
- Pisahkan waktu setup, alokasi, kompilasi, dan kernel.

9.2.3 Line Profiler

Line profiler menunjukkan baris yang mahal dalam fungsi Python. Ia paling berguna setelah fungsi bottleneck diketahui.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Lakukan warm-up dan laporkan median serta variasi waktu.
- Pisahkan waktu setup, alokasi, kompilasi, dan kernel.

9.2.4 Sampling Profiler

Sampling profiler menginterupsi proses secara periodik dan memiliki overhead rendah. Ia dapat melihat waktu di native code.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Lakukan warm-up dan laporkan median serta variasi waktu.
- Pisahkan waktu setup, alokasi, kompilasi, dan kernel.

9.2.5 perf

Linux perf menghubungkan program dengan counter hardware, cache miss, branch miss, dan stack native. Interpretasi memerlukan simbol dan build yang sesuai.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai

yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.3 Optimasi Tingkat Python

Optimasi tingkat Python mengurangi dispatch, alokasi, dan akses objek. Setelah sebagian besar waktu berada di native kernel, optimasi perlu berpindah ke algoritma atau backend.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

9.3.1 Modul dis

dis menampilkan bytecode dan membantu memahami loop, lookup atribut, serta pembuatan objek.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.3.2 Kode Python yang Ramah Kinerja

Kurangi kerja di loop terdalam, simpan lookup lokal, hindari alokasi berulang, dan gunakan struktur data yang sesuai.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Lakukan warm-up dan laporkan median serta variasi waktu.
- Pisahkan waktu setup, alokasi, kompilasi, dan kernel.

9.3.3 Memanfaatkan Operasi Tensor

einsum, matmul, dan fungsi BLAS menggabungkan banyak operasi dalam kernel teroptimasi. Jalur kontraksi perlu dievaluasi.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

9.3.4 Mengoptimalkan Efisiensi Indexing Tensor

Slicing kontigu dan pengurutan sumbu yang tepat mengurangi gather/scatter. Transpose yang hanya view dapat tetap memicu salinan di kernel berikutnya.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

9.4 Mengompilasi Kode Python

Compiler khusus Python memerlukan subset kode yang dapat dianalisis. Tipe yang stabil dan kontrol efek samping memperbesar peluang optimasi.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

9.4.1 Numba

Numba mengompilasi fungsi numerik bertipe stabil melalui LLVM. Mode nopython harus menjadi sasaran utama.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.4.2 Cython

Cython efektif untuk loop kompleks dan integrasi C, tetapi membutuhkan build serta anotasi tipe.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.4.3 Pythran

Pythran mengompilasi subset Python numerik dan dapat mengoptimalkan ekspresi array. Kode harus mengikuti subset yang didukung.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.4.4 Perbandingan Cython, Pythran, dan Numba

Perbandingan mempertimbangkan perubahan kode, waktu build, portabilitas, debugging, dan pola kernel, bukan hanya waktu runtime.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.5 Optimasi dengan Bahasa Terkompilasi

Bahasa terkompilasi memberi kontrol terhadap SIMD, pointer, dan memory layout. Biaya pengembangan serta potensi bug memori harus diperhitungkan.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

9.5.1 Kompiler GCC

Flag optimasi, vectorization report, fast-math, dan target arsitektur memengaruhi kinerja serta semantik floating-point.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

9.6 Optimasi I/O

I/O sering menjadi bottleneck tersembunyi pada checkpoint dan tensor disk-backed. Mengurangi jumlah transfer biasanya lebih efektif daripada mempercepat satu operasi baca.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

9.6.1 Tata Letak Penyimpanan

Layout menentukan apakah pembacaan berlangsung berurutan atau melompat. Chunking dan pengelompokan data harus mengikuti pola kerja.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.6.2 Kompresi Data

Kompresi menukar CPU dengan bandwidth dan ruang. Data floating-point acak mungkin sulit dikompresi tanpa transformasi atau toleransi lossy.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

9.6.3 Menumpangtindihkan Komputasi dan I/O

Prefetch, double buffering, dan thread I/O dapat menyembunyikan latensi jika komputasi dan transfer menggunakan sumber daya berbeda.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.7 Precomputation dan Memoization

Pra-hitung menukar waktu dengan memori dan kompleksitas invalidasi. Cache harus memiliki kunci yang mewakili semua input yang memengaruhi hasil.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

9.7.1 LRU Cache

LRU cache menyimpan hasil panggilan terbaru berdasarkan argumen hashable. Cache tidak cocok bila fungsi bergantung pada keadaan tersembunyi.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.7.2 Pemrograman Fungsional

Fungsi murni memudahkan cache, paralelisme, dan pengujian karena keluaran hanya bergantung pada input.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.7.3 Pemrograman Dinamis

Dynamic programming menyimpan submasalah yang berulang. Relasi rekurensi integral Gaussian adalah contoh alami.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

9.8 Optimasi dengan Lazy Evaluation

Lazy evaluation membangun rencana sebelum mengeksekusi, sehingga operasi dapat digabung atau dihindari. Namun error muncul lebih lambat dan penggunaan memori dapat sulit diprediksi.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 9.8 optimasi dengan lazy evaluation akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

Praktikum Terpandu

Memprofilkan evaluasi fungsi basis pada grid, memindahkan loop ke NumPy dan Numba, lalu membandingkan waktu serta ketepatan.

Contoh kode Bab 9

```
import numpy as np
from numba import njit

@njit(cache=True)
def gaussian_sum(x: np.ndarray, alpha: np.ndarray, coeff: np.ndarray) ->
np.ndarray:
    out = np.zeros_like(x)
    for p in range(alpha.size):
        for i in range(x.size):
            out[i] += coeff[p] * np.exp(-alpha[p] * x[i] * x[i])
    return out

x = np.linspace(-8.0, 8.0, 200_000)
a = np.array([0.2, 0.8, 3.0, 12.0])
c = np.array([0.1, 0.3, 0.4, 0.2])
y = gaussian_sum(x, a, c)
print(y.min(), y.max())
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Optimasi mengikuti urutan: ukur, modelkan, ubah algoritma, lalu mikro-optimasi.
- Kompilasi membantu kernel numerik bertipe stabil, bukan seluruh aplikasi secara otomatis.
- Layout data, alokasi, dan I/O sering sama pentingnya dengan jumlah operasi aritmetika.

Latihan dan Refleksi

25. Buat model biaya untuk kontraksi empat indeks dan identifikasi dimensi dominan.
26. Jelaskan kapan sampling profiler lebih tepat daripada line profiler.
27. Rancang benchmark adil untuk membandingkan NumPy dan Numba.

Proyek mini: Memprofilkan evaluasi fungsi basis pada grid, memindahkan loop ke NumPy dan Numba, lalu membandingkan waktu serta ketepatan.

BAB 10
KOMPUTASI PARALEL

Paralelisme dapat dilakukan dengan thread, proses, coroutine, OpenMP, atau MPI. Setiap model memiliki ruang memori, mekanisme sinkronisasi, dan biaya komunikasi yang berbeda. Bab ini menekankan pemilihan model berdasarkan dependensi data dan granularitas tugas.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- membedakan concurrency dari parallelism
- menggunakan thread dan proses tanpa race condition
- menerapkan producer-consumer dan pipeline
- memahami coroutine serta asyncio untuk tugas I/O
- merancang program MPI SPMD dan memanfaatkan shared memory

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

10.1 Multithreading

Thread berbagi memori dan cocok untuk I/O atau kernel native yang melepas GIL. Data bersama memerlukan protokol sinkronisasi.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

10.1.1 Modul threading

Thread berbagi ruang memori dan ringan untuk koordinasi, tetapi bytecode Python dibatasi GIL pada interpreter standar. Native kernel yang melepas GIL tetap dapat berjalan paralel.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.

- Ukur komunikasi atau transfer selain komputasi.

10.1.2 ThreadPoolExecutor

ThreadPoolExecutor menyediakan API futures untuk tugas I/O atau native code. Jumlah worker perlu disesuaikan dengan threading internal BLAS agar tidak oversubscription.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

10.2 Lock dan Sinkronisasi Thread

Sinkronisasi menjaga invariant, tetapi menambah serialisasi dan risiko deadlock. Desain berbasis message passing sering lebih mudah ditinjau.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

10.2.1 Mutex

Mutex melindungi invariant yang melibatkan data bersama. Critical section harus kecil dan urutan lock konsisten untuk menghindari deadlock.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

10.2.2 Event dan Condition Variable

Event menyatakan keadaan satu arah, sedangkan condition menggabungkan lock dengan penantian terhadap predikat. Selalu uji predikat dalam loop karena wakeup dapat bersifat spurious.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai

yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

10.3 Model Producer-Consumer

Producer-consumer memisahkan laju produksi dan konsumsi melalui queue. Backpressure mencegah memori bertumbuh tanpa batas.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 10.3 model producer-consumer akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

10.4 Eksekutor Pipeline

Pipeline menempatkan tahap berbeda secara tumpang tindih, misalnya membaca blok, menghitung integral, lalu mengontraksi. Throughput dibatasi tahap paling lambat.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 10.4 eksekutor pipeline akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

10.5 Program Asinkron

Async efektif ketika tugas sering menunggu I/O. Ia tidak secara otomatis mempercepat komputasi CPU.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

10.5.1 Kesamaan Asynchronous Programming dan Lazy Evaluation

Keduanya menunda kerja sampai diperlukan. Perbedaannya, async mengatur penantian dan interleaving, sedangkan lazy berfokus pada kapan nilai dievaluasi.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

10.5.2 Coroutine dan asyncio

Coroutine melepaskan kontrol pada titik await sehingga satu thread dapat menangani banyak operasi I/O. Fungsi CPU berat tetap harus dipindahkan ke thread, proses, atau layanan lain.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

10.6 Multiprocessing

Proses menghindari GIL untuk bytecode CPU-bound dan memberi isolasi. Startup serta serialisasi menjadi biaya utama.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

10.6.1 Komunikasi Data dalam multiprocessing.Process

Process memiliki memori terpisah. Pipe, Queue, shared memory, atau file diperlukan untuk komunikasi.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

10.6.2 Komunikasi Data dalam ProcessPoolExecutor

Argumen dan hasil biasanya diserialkan. Tugas harus cukup besar agar biaya serialisasi dan startup teramortisasi.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

10.6.3 Lock dalam Program Multiprocessing

Lock antarproses bekerja melalui primitive OS dan lebih mahal daripada lock thread. Desain tanpa berbagi keadaan sering lebih sederhana.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

10.7 Komunikasi Antarproses

Pilihan pipe, queue, socket, shared memory, atau memory-mapped file menentukan semantik dan biaya. Kepemilikan buffer harus jelas.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 10.7 komunikasi antarproses akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

10.8 OpenMP

OpenMP menambahkan parallel region dan work-sharing pada kode C/C++/Fortran. Python biasanya berinteraksi melalui ekstensi yang melepas GIL.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 10.8 openmp akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

10.9 MPI

MPI menyediakan komunikasi eksplisit antarproses dan merupakan standar utama untuk distributed-memory HPC. Kinerja bergantung pada volume, pola, dan sinkronisasi komunikasi.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

10.9.1 Konvensi Penamaan dalam MPI4Py

MPI4Py menyediakan metode Pythonic dan binding buffer huruf kapital. Binding buffer menghindari pickle dan cocok untuk ndarray kontigu.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

10.9.2 Program SPMD MPI dalam Python

Semua rank menjalankan program yang sama dengan data berbeda. Rank dan size mengontrol pembagian kerja.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

10.9.3 Mengintegrasikan MPI ke Program Serial

Mulai dari pembagian independen, pertahankan jalur serial, dan kumpulkan hasil pada batas yang jelas. Hindari menyebarkan logika rank ke seluruh kode.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

10.9.4 Shared Memory

MPI shared-memory window memungkinkan rank pada node yang sama berbagi buffer. Sinkronisasi dan layout tetap harus dikelola.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Menjalankan pemindaian energi paralel dengan ProcessPoolExecutor, membatasi thread BLAS per proses, dan mengumpulkan hasil secara deterministik.

Contoh kode Bab 10

```
from concurrent.futures import ProcessPoolExecutor
import os

def energy_task(r: float) -> tuple[float, float]:
    # Dalam produksi, atur OMP_NUM_THREADS sebelum impor pustaka BLAS.
    energy = -1.0 + 0.65 * (r - 0.74)**2
    return r, energy

if __name__ == "__main__":
    distances = [0.5 + 0.02*i for i in range(100)]
    with ProcessPoolExecutor(max_workers=4) as pool:
        results = list(pool.map(energy_task, distances, chunksize=5))
    results.sort(key=lambda item: item[0])
    print(min(results, key=lambda item: item[1]))
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Model paralel dibedakan oleh ruang memori, komunikasi, dan sinkronisasi.
- Granularitas tugas harus cukup besar untuk menutup overhead penjadwalan dan transfer.
- Oversubscription antara proses, thread Python, dan thread BLAS harus dicegah.

Latihan dan Refleksi

28. Jelaskan kapan ThreadPoolExecutor dapat memberi percepatan meskipun ada GIL.
29. Rancang producer-consumer untuk membaca integral dari disk sambil mengontraksikannya.
30. Tulis strategi pembagian grid kepada rank MPI dan pengumpulan hasil.

Proyek mini: Menjalankan pemindaian energi paralel dengan ProcessPoolExecutor, membatasi thread BLAS per proses, dan mengumpulkan hasil secara deterministik.

BAB 11
PEMROGRAMAN GPU

GPU menawarkan throughput tinggi melalui paralelisme masif, tetapi percepatan hanya tercapai jika transfer data, bentuk kernel, occupancy, dan intensitas aritmetika sesuai. Bab ini memulai dari pustaka tingkat tinggi dan bergerak menuju kernel khusus.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- menyiapkan runtime GPU dan memverifikasi kompatibilitas
- menggunakan CuPy, PyTorch, dan JAX sebagai lapisan portabel
- memahami pinned memory dan CUDA stream
- menulis kernel CUDA atau Numba yang sederhana
- menentukan kapan kernel khusus layak dikembangkan

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

11.1 Mengonfigurasi Lingkungan Runtime GPU

Runtime GPU mencakup driver, toolkit, library, dan paket Python yang kompatibel. Pemeriksaan device dan versi harus dilakukan sebelum benchmark.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 11.1 mengonfigurasi lingkungan runtime gpu akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

11.2 Optimasi Independen Arsitektur

Lapisan array tingkat tinggi memungkinkan kode berpindah antarperangkat. Pola operasi tetap harus menghindari transfer dan kernel kecil beruntun.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

11.2.1 CuPy

CuPy meniru sebagian besar API NumPy dan menjalankan array pada GPU. Transfer host-device harus dibuat eksplisit dan diminimalkan.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

11.2.2 PyTorch

PyTorch menggabungkan tensor GPU, autograd, dan kernel teroptimasi. Tipe, device, dan mode gradien perlu dikelola dengan konsisten.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

11.2.3 JAX

JAX menempatkan array pada accelerator melalui XLA. JIT dan vmap dapat menggabungkan operasi, tetapi recompilation terjadi bila bentuk atau static argument berubah.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.

- Dokumentasikan asumsi dan batas penggunaan.

11.3 Optimasi Sadar Arsitektur

Optimasi sadar arsitektur mempertimbangkan memory hierarchy, stream, occupancy, dan coalescing. Ia memberi hasil terbesar setelah algoritma dasar sudah cocok untuk GPU.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil. Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

11.3.1 Pinned Memory

Pinned memory memungkinkan transfer DMA yang lebih efisien dan asinkron. Penggunaannya berlebihan dapat mengurangi memori pageable sistem.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

11.3.2 CUDA Stream

Stream mengurutkan operasi dan memungkinkan overlap antarkegiatan independen. Dependensi harus dinyatakan dengan event atau sinkronisasi.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

11.4 Kernel GPU Khusus dalam Python

Kernel khusus dapat menggabungkan operasi dan mengurangi array sementara. Manfaatnya harus melebihi biaya pengembangan dan validasi.

Evaluasi teknik tidak cukup dengan satu angka waktu. Catat ukuran data, jumlah thread atau proses, backend, warm-up, alokasi sementara, dan ketepatan hasil.

Percepatan yang mengubah toleransi atau urutan operasi floating-point harus dilaporkan sebagai trade-off numerik.

11.4.1 Kernel dalam Kode CUDA

Kernel CUDA memetakan thread dan block ke indeks data. Coalesced access dan penghindaran divergence menjadi perhatian utama.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

11.4.2 Numba CUDA JIT

Numba CUDA mengompilasi fungsi Python terbatas menjadi kernel. Ia baik untuk prototipe tanpa toolchain binding terpisah.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

11.4.3 Kernel Khusus CuPy

RawKernel dan ElementwiseKernel memungkinkan integrasi CUDA khusus sambil mempertahankan pengelolaan array CuPy.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

11.4.4 Mengintegrasikan Kernel CUDA dengan ctypes

Driver atau runtime API dapat dipanggil melalui ctypes, tetapi pengelolaan pointer dan context menjadi lebih kompleks daripada memakai binding khusus.

Dampak kinerja harus dibaca bersama correctness. Gunakan input representatif, ulangi pengukuran, dan periksa bahwa implementasi cepat menghasilkan nilai yang setara dalam toleransi. Perubahan layout atau paralelisme dapat mengubah urutan penjumlahan dan menghasilkan selisih kecil yang tetap perlu dijelaskan.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

Praktikum Terpandu

Memindahkan evaluasi fungsi basis berbasis grid dari NumPy ke CuPy dan mengukur waktu transfer serta waktu kernel secara terpisah.

Contoh kode Bab 11

```
import cupy as cp

x = cp.linspace(-8.0, 8.0, 2_000_000, dtype=cp.float64)
alpha = cp.asarray([0.2, 0.8, 3.0, 12.0])
coeff = cp.asarray([0.1, 0.3, 0.4, 0.2])

# Broadcasting menghasilkan matriks sementara; untuk kasus besar pertimbangkan
kernel fused.
values = cp.exp(-alpha[:, None] * x[None, :]**2)
y = coeff @ values
cp.cuda.Stream.null.synchronize()
print(float(y.max().get()))
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- GPU menguntungkan bila data tinggal cukup lama di device dan kernel memiliki paralelisme serta intensitas aritmetika tinggi.
- API tingkat tinggi mempercepat pengembangan, sedangkan kernel khusus menambah beban pemeliharaan.
- Pengukuran harus memisahkan transfer, kompilasi, dan eksekusi kernel.

Latihan dan Refleksi

31. Jelaskan mengapa operasi GPU kecil dapat lebih lambat daripada CPU.
32. Hitung ukuran transfer untuk tensor 1000×1000 complex128.
33. Rancang eksperimen untuk mengukur overlap transfer dan komputasi menggunakan dua stream.

Proyek mini: Memindahkan evaluasi fungsi basis berbasis grid dari NumPy ke CuPy dan mengukur waktu transfer serta waktu kernel secara terpisah.

BAGIAN 3

APLIKASI KIMIA KUANTUM DENGAN PYTHON

Bagian ini menerapkan teknik sebelumnya pada integral, mean-field, FCI, coupled cluster, serta sifat dan turunan molekuler.

BAB 12
EVALUASI INTEGRAL

Integral atas fungsi basis Gaussian merupakan inti banyak metode struktur elektronik. Bab ini membahas representasi basis, relasi rekurensi, pemrograman dinamis, penghilangan cabang, kompilasi, kuadratur numerik, transformasi Fourier, dan transformasi indeks.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- merepresentasikan shell dan primitive Gaussian secara terstruktur
- menerapkan relasi rekurensi integral secara iteratif
- menggunakan memoization dan code generation untuk spesialisasi kernel
- memahami kuadratur Gaussian dan transformasi Fourier untuk integral numerik
- melakukan transformasi integral dengan memperhatikan biaya dan memori

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

12.1 Evaluasi Integral Analitik untuk Orbital Tipe Gaussian

Gaussian product theorem mengubah hasil kali dua Gaussian menjadi Gaussian baru sehingga integral analitik dapat direduksi menjadi kasus dasar dan relasi rekurensi.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

12.1.1 Struktur Data Basis GTO

Basis GTO disusun sebagai atom, shell, momentum sudut, eksponen, koefisien kontraksi, dan normalisasi. Struktur data harus mendukung grouping shell serta akses kontigu.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

12.1.2 Tipe Dasar Integral GTO Analitik

Integral overlap, kinetik, tarik inti, dan repulsi elektron berbagi komponen Gaussian product theorem dan integral bantu.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

12.1.3 Relasi Rekurensi dan Implementasi Pemrograman Dinamis

Relasi rekurensi membangun momentum sudut tinggi dari kasus dasar. Dynamic programming menyimpan nilai perantara agar cabang yang sama tidak dihitung berulang.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

12.1.4 Mengubah Rekursi menjadi Iterasi

Iterasi menghindari overhead call dan memberi kontrol atas urutan pengisian tabel. Urutan harus menghormati dependensi rekurensi.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

12.1.5 Menghilangkan Cabang dan Menyesuaikan Iterasi

Spesialisasi untuk momentum sudut tertentu dapat menghapus if di loop terdalam. Padding atau loop terpisah sering lebih cepat daripada cabang tidak terprediksi.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

12.1.6 Optimasi dengan Kompilasi Numba JIT

Numba efektif untuk loop aritmetika dengan array bertipe tetap. Alokasi di kernel harus diminimalkan.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

12.1.7 Optimasi dengan Kompilasi Cython

Cython memberikan kontrol tipe dan pointer yang lebih rinci untuk kernel integral.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Identifikasi data bersama dan titik sinkronisasi.
- Ukur komunikasi atau transfer selain komputasi.

12.1.8 Optimasi dengan Teknik Metaprogramming

Generator dapat menghasilkan kernel khusus shell quartet, unroll loop, dan ekspresi perantara yang telah disederhanakan.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

12.1.9 Benchmark Kinerja

Benchmark integral harus menyertakan distribusi shell realistis, screening, cache, dan verifikasi nilai.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Lakukan warm-up dan laporkan median serta variasi waktu.
- Pisahkan waktu setup, alokasi, kompilasi, dan kernel.

12.1.10 Faktor Kinerja Lain

Vectorization, alignment, batching shell, symmetry, screening, dan parallel reduction sama pentingnya dengan bahasa implementasi.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Lakukan warm-up dan laporkan median serta variasi waktu.
- Pisahkan waktu setup, alokasi, kompilasi, dan kernel.

12.2 Integrasi Numerik

Integrasi numerik diperlukan untuk operator atau fungsional yang tidak dievaluasi sepenuhnya secara analitik. Grid dan bobot harus memenuhi akurasi serta simetri.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

12.2.1 Kuadratur Gaussian

Kuadratur Gaussian memilih node dan bobot agar polinom sampai derajat tertentu terintegrasi tepat terhadap fungsi bobot.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

12.2.2 Menghasilkan Akar dan Bobot Kuadratur

Akar dan bobot dapat diperoleh dari polinom ortogonal atau masalah eigen tridiagonal. Stabilitas penting pada orde tinggi.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

12.2.3 Mendekati Kuadratur

Pendekatan atau tabel pra-hitung mempercepat evaluasi, tetapi galat harus dikontrol di seluruh rentang parameter.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

12.2.4 Integrasi Numerik dengan Transformasi Fourier

Konvolusi atau operator diferensial tertentu lebih mudah dihitung di ruang Fourier. Periodisitas dan aliasing harus diperhatikan.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

12.3 Transformasi Integral

Transformasi AO-ke-MO mengontraksikan setiap indeks dengan koefisien orbital. Urutan transformasi bertahap menghindari biaya dan memori yang tidak perlu.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 12.3 transformasi integral akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

Praktikum Terpandu

Membangun integral overlap satu dimensi untuk Gaussian kartesian dengan relasi rekurensi, lalu memvalidasinya terhadap kuadratur numerik.

Contoh kode Bab 12

```
from functools import lru_cache
import math

@lru_cache(maxsize=None)
def overlap_1d(i: int, j: int, a: float, b: float, A: float, B: float) ->
float:
    p = a + b
    P = (a*A + b*B) / p
    mu = a*b/p
    base = math.sqrt(math.pi/p) * math.exp(-mu*(A-B)**2)
    if i == 0 and j == 0:
        return base
    if i > 0:
        term1 = (P-A) * overlap_1d(i-1, j, a, b, A, B)
        term2 = (i-1)/(2*p) * overlap_1d(i-2, j, a, b, A, B) if i > 1 else 0.0
        term3 = j/(2*p) * overlap_1d(i-1, j-1, a, b, A, B) if j > 0 else 0.0
        return term1 + term2 + term3
    term1 = (P-B) * overlap_1d(i, j-1, a, b, A, B)
    term2 = (j-1)/(2*p) * overlap_1d(i, j-2, a, b, A, B) if j > 1 else 0.0
    return term1 + term2

print(overlap_1d(1, 0, 0.8, 1.1, 0.0, 0.7))
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Representasi basis dan urutan shell menentukan kemudahan optimasi kernel integral.
- Relasi rekurensi cocok untuk dynamic programming, iterasi, dan spesialisasi kode.

- Benchmark harus memadukan kebenaran numerik, screening, dan pola workload realistis.

Latihan dan Refleksi

34. Turunkan kasus dasar integral overlap dua Gaussian s.
35. Ubah fungsi rekurensif overlap menjadi tabel iteratif.
36. Jelaskan empat tahap transformasi integral AO ke MO dan biaya asimtotiknya.

Proyek mini: Membangun integral overlap satu dimensi untuk Gaussian kartesian dengan relasi rekurensi, lalu memvalidasinya terhadap kuadratur numerik.

BAB 13
METODE MEAN-FIELD

Metode mean-field mengubah masalah banyak elektron menjadi iterasi self-consistent. Bab ini menyatukan persamaan Roothaan-Hall, pembentukan matriks Coulomb dan exchange, integrasi DFT, DIIS, desain kelas, serta teknik percepatan seperti density fitting dan tebakan awal.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- menulis loop SCF dengan kriteria konvergensi yang jelas
- membangun dan memeriksa matriks Coulomb serta exchange
- mengintegrasikan densitas dan potensial exchange-correlation pada grid
- menerapkan DIIS secara stabil
- merancang kelas metode elektronik yang dapat diperluas

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

13.1 Program Iterasi Self-Consistency

SCF membangun Fock dari densitas, menyelesaikan eigenproblem, memperbarui densitas, dan mengulang sampai fixed-point. Konvergensi energi saja belum selalu menjamin densitas stabil.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 13.1 program iterasi self-consistency akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

13.2 Matriks Coulomb dan Exchange

J mewakili interaksi Coulomb rata-rata, sedangkan K berasal dari antisimetri fermion. Konstruksi keduanya mendominasi HF konvensional.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

13.2.1 Screening Integral untuk Matriks Coulomb dan Exchange

Batas Schwarz dan estimator lain menghindari shell quartet yang kontribusinya di bawah ambang. Ambang screening memengaruhi biaya dan reproduksibilitas.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

13.2.2 J-engine

J-engine mengorganisasi evaluasi integral dan kontraksi densitas secara langsung. Desainnya memanfaatkan symmetry serta batching untuk locality.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

13.3 Integral untuk Fungsional Exchange-Correlation

DFT mengevaluasi densitas dan turunannya pada grid, memanggil fungsional XC, lalu mengintegrasikan potensial kembali ke basis AO.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.

- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 13.3 integral untuk fungsional exchange-correlation akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

13.4 DIIS

DIIS membangun kombinasi linear iterasi sebelumnya yang meminimalkan error. Pemilihan error vector dan ukuran subspace menentukan stabilitas.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 13.4 diis akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

13.5 Desain Kelas Python untuk Metode Mean-Field

API mean-field sebaiknya memisahkan molekul, integral, keadaan iterasi, dan opsi. Ekstensi baru harus dapat mengganti komponen tanpa menyalin keseluruhan algoritma.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

13.5.1 Kelas Mean-Field Baru melalui Pewarisan Kelas

Pewarisan dapat mengubah bagian tertentu seperti pembentukan Fock atau okupansi. Kontrak metode dasar harus stabil.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

13.5.2 Kelas Mean-Field melalui Visitor Pattern

Visitor memisahkan operasi tambahan dari kelas objek. Pola ini berguna untuk analisis atau transformasi yang tidak ingin memperbesar hierarki inti.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

13.5.3 Patch Dinamis pada Objek Mean-Field

Patch dinamis memungkinkan eksperimen cepat, tetapi sulit dilacak. Simpan referensi fungsi asli dan catat perubahan dalam provenance.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

13.5.4 Kelas Mean-Field Dinamis

Kelas dapat dibentuk dari kombinasi fitur saat runtime. Nama kelas, urutan MRO, dan serialisasi perlu dipertimbangkan.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

13.6 Mempercepat Perhitungan Mean-Field

Percepatan datang dari pengurangan rank integral, screening, initial guess, dan convergence accelerator. Setiap pendekatan memiliki trade-off galat dan memori.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

13.6.1 Dekomposisi Cholesky dan Resolution of Identity

CD dan RI memfaktorkan tensor empat indeks menjadi faktor berderajat lebih rendah, menurunkan penyimpanan serta biaya kontraksi.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

13.6.2 Meningkatkan Konvergensi dengan DIIS

DIIS mengekstrapolasi Fock atau error vector dari riwayat iterasi. Regularisasi dan pengelolaan subspace mencegah sistem linear menjadi singular.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

13.6.3 Meningkatkan Tebakan Awal

Tebakan atom, core Hamiltonian, SAD, atau hasil geometri sebelumnya dapat memperpendek iterasi dan mengarahkan solusi ke keadaan yang diinginkan.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Menjalankan RHF dengan PySCF, mengambil matriks overlap dan Fock, memeriksa residual komutator, serta membandingkan beberapa initial guess.

Contoh kode Bab 13

```
from pyscf import gto, scf
import numpy as np

mol = gto.M(
    atom="H 0 0 0; H 0 0 0.74",
    basis="sto-3g",
    unit="Angstrom",
    verbose=0,
)
mf = scf.RHF(mol)
mf.conv_tol = 1e-10
energy = mf.kernel()

S = mf.get_ovlp()
D = mf.make_rdm1()
F = mf.get_fock(dm=D)
residual = F @ D @ S - S @ D @ F
print("E(RHF) =", energy)
print("||FDS-SDF|| =", np.linalg.norm(residual))
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- SCF adalah masalah fixed-point dengan kriteria konvergensi energi dan densitas.
- Pembentukan J/K atau potensial XC biasanya mendominasi biaya.
- DIIS, RI/CD, dan tebakan awal memperbaiki konvergensi serta skala komputasi.

Latihan dan Refleksi

37. Tuliskan pseudocode loop RHF lengkap dengan orthogonalization dan damping.
38. Jelaskan makna residual komutator FDS-SDF.
39. Bandingkan direct SCF, density fitting, dan penyimpanan integral penuh.

Proyek mini: Menjalankan RHF dengan PySCF, mengambil matriks overlap dan Fock, memeriksa residual komutator, serta membandingkan beberapa initial guess.

BAB 14
POST-HARTREE-FOCK I: FULL CONFIGURATION INTERACTION

Full configuration interaction menjadi acuan eksak dalam basis orbital terbatas, tetapi ruang determinannya bertumbuh kombinatorial. Bab ini membahas representasi bit-string, Davidson, algoritma direct CI, kontraksi tensor, efisiensi memori, dan paralelisme.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- menghitung ukuran ruang determinan
- merepresentasikan okupansi dengan bit-string
- menerapkan Davidson untuk eigenpair terendah
- menghindari pembentukan Hamiltonian penuh melalui direct CI
- mengelola distribusi vektor CI dan kontraksi secara paralel

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

14.1 Teori Full Configuration Interaction

FCI mengekspansi fungsi gelombang pada seluruh determinan yang mungkin dalam ruang orbital. Energinya variational dan menjadi batas dalam basis tersebut.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 14.1 teori full configuration interaction akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

14.2 Representasi String

Bit-string menyimpan okupansi orbital dalam integer dan memungkinkan operasi bit cepat untuk menghitung eksitasi serta fase fermion.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 14.2 representasi string akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

14.3 Diagonalisasi Davidson

Davidson membangun subruang dari residual dan menggunakan diagonal sebagai preconditioner. Metode ini efektif untuk eigenvalue ekstrem pada matriks besar.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 14.3 diagonalisasi davidson akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

14.4 Algoritma Direct CI

Direct CI menghitung aksi Hamiltonian pada vektor tanpa menyimpan matriks penuh. Tabel link dan faktorisasi sigma vector menentukan kinerja.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 14.4 algoritma direct ci akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

14.5 Mengoptimalkan Kontraksi Tensor

Kontraksi CI perlu menggabungkan BLAS, blok simetri, dan reordering agar akses memori teratur.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 14.5 mengoptimalkan kontraksi tensor akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

14.6 Optimasi Efisiensi Memori

Vektor CI dapat jauh lebih besar daripada cache atau RAM. Blocking, distribusi, dan tipe data harus direncanakan.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 14.6 optimasi efisiensi memori akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

14.7 Komputasi Paralel

Paralelisme membagi string, blok Hamiltonian, atau kontribusi integral. Komunikasi reduksi dapat menjadi bottleneck.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 14.7 komputasi paralel akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

Praktikum Terpandu

Menggunakan solver FCI PySCF untuk H2 dan membandingkan energi dengan RHF; kemudian menghitung dimensi ruang determinan untuk beberapa jumlah orbital.

Contoh kode Bab 14

```
from math import comb
from pyscf import gto, scf, fci

mol = gto.M(atom="H 0 0 0; H 0 0 1.4", basis="sto-3g", unit="Bohr", verbose=0)
mf = scf.RHF(mol).run()
solver = fci.FCI(mol, mf.mo_coeff)
e_fci, ci = solver.kernel()

norb = mf.mo_coeff.shape[1]
na = nb = mol.nelectron // 2
dim = comb(norb, na) * comb(norb, nb)
print("E RHF:", mf.e_tot)
print("E FCI:", e_fci)
print("dimensi determinan:", dim)
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- FCI eksak dalam basis terbatas tetapi tumbuh kombinatorial.
- Bit-string dan tabel eksitasi membuat operasi determinan efisien.
- Davidson dan direct CI menghindari penyimpanan Hamiltonian penuh.

Latihan dan Refleksi

40. Hitung dimensi FCI untuk 10 orbital ruang dan 10 elektron singlet.
41. Jelaskan peran preconditioner diagonal pada Davidson.
42. Rancang pembagian vektor CI untuk beberapa proses tanpa duplikasi berlebihan.

Proyek mini: Menggunakan solver FCI PySCF untuk H2 dan membandingkan energi dengan RHF; kemudian menghitung dimensi ruang determinan untuk beberapa jumlah orbital.

BAB 15
POST-HARTREE-FOCK II: COUPLED CLUSTER

Coupled cluster menggunakan ansatz eksponensial dan persamaan amplitudo nonlinear. Bab ini menampilkan CCD sebagai contoh desain kernel kontraksi, pemindahan data, prefetch, dan pembuatan persamaan secara simbolik.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- menjelaskan ansatz eksponensial dan connectedness
- menata amplitudo serta integral untuk program CCD
- mengurangi transfer data pada kontraksi besar
- menggunakan prefetch untuk menumpangtindihkan I/O
- mewakili operator dan simetri dalam sistem aljabar simbolik

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

15.1 Teori Coupled Cluster

Ansatz exp(T)|Φ0⟩ menghasilkan kontribusi connected dan size extensivity. Persamaan diperoleh dengan proyeksi transformed Hamiltonian.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 15.1 teori coupled cluster akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

15.2 Program CCD

CCD menyimpan amplitudo eksitasi ganda dan memperbaruinya melalui residual nonlinear. Denominator orbital memberi initial guess dan preconditioner.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 15.2 program ccd akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

15.3 Mengoptimalkan Pemindahan Data

Kontraksi skala tinggi sering dibatasi bandwidth. Reuse perantara, blocking, dan tata letak mengurangi pemindahan.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 15.3 mengoptimalkan pemindahan data akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

15.4 Kompilasi Just-in-Time untuk I/O Readahead

Prefetch dapat dihasilkan atau dikompilasi untuk pola blok yang diketahui. Tujuannya mengisi buffer sebelum kernel membutuhkan data.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 15.4 kompilasi just-in-time untuk i/o readahead akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

15.5 Pemrograman Simbolik untuk Teori Coupled Cluster

Aljabar operator menghasilkan banyak suku dengan simetri dan tanda fermion. Sistem simbolik membantu kanonisasi, penyederhanaan, dan code generation.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

15.5.1 Mendefinisikan Elemen Simbolik Fundamental

Indeks orbital, tensor, operator, koefisien, dan domain spin perlu mempunyai identitas serta aturan kanonik.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

15.5.2 Menerapkan Aturan Komputasi

Aturan kontraksi, Wick, dan tanda fermion diterapkan sebagai transformasi pada ekspresi.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel,

serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

15.5.3 Mengevaluasi Ekspresi Secara Simbolik

Evaluasi menggabungkan suku ekuivalen, mengontraksikan delta, dan menormalkan urutan indeks.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

15.5.4 Menghasilkan Persamaan CCD Secara Simbolik

Proyeksi persamaan amplitudo menghasilkan daftar kontraksi yang kemudian diterjemahkan menjadi einsum atau kernel khusus.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

15.5.5 Menerapkan Simetri Permutasi pada Tensor CCD

Antisimetri amplitudo dan integral mengurangi jumlah elemen unik serta suku. Kanonisasi indeks mencegah duplikasi.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

Praktikum Terpandu

Menjalankan CCSD pada molekul kecil, mengambil energi korelasi, dan memeriksa sensitivitas terhadap ambang konvergensi.

Contoh kode Bab 15

```
from pyscf import gto, scf, cc

mol = gto.M(
    atom="O 0 0 0; H 0 -0.757 0.587; H 0 0.757 0.587",
    basis="sto-3g",
    unit="Angstrom",
    verbose=0,
)
mf = scf.RHF(mol).run(conv_tol=1e-11)
mycc = cc.CCSD(mf)
mycc.conv_tol = 1e-9
e_corr, t1, t2 = mycc.kernel()
print("E korelasi CCSD:", e_corr)
print("E total CCSD:", mf.e_tot + e_corr)
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Ansatz eksponensial menghasilkan size extensivity dan persamaan nonlinear.
- Kinerja CCD/CCSD ditentukan oleh kontraksi tensor, layout, dan transfer data.
- Pemrograman simbolik mengurangi kesalahan derivasi serta memungkinkan code generation.

Latihan dan Refleksi

43. Jelaskan perbedaan size consistency dan size extensivity.
44. Identifikasi kontraksi dominan dalam persamaan CCD dan skala formalnya.
45. Rancang representasi simbolik tensor yang menyimpan simetri pertukaran indeks.

Proyek mini: Menjalankan CCSD pada molekul kecil, mengambil energi korelasi, dan memeriksa sensitivitas terhadap ambang konvergensi.

BAB 16
SIFAT MOLEKULER

Energi saja tidak cukup; kimia memerlukan dipol, gradien, respons, dan turunan orde tinggi. Bab ini membahas operator satu partikel, gradien analitik RHF, finite difference, orbital orde pertama, solver Krylov, serta diferensiasi otomatis dan implisit.

Capaian pembelajaran: Setelah menyelesaikan bab ini, pembaca diharapkan mampu:

- menghitung nilai harapan operator satu partikel
- menjelaskan komponen gradien analitik
- menggunakan finite difference sebagai validasi, bukan pengganti otomatis
- memecahkan persamaan respons dengan metode Krylov
- menerapkan diferensiasi otomatis pada prosedur iteratif secara efisien

Prasyarat: Dasar Python, aljabar linear, notasi indeks, dan pengantar kimia kuantum. Beberapa contoh memerlukan NumPy, SciPy, Pandas, Matplotlib, PySCF, SymPy, Numba, JAX, CuPy, Dask, atau h5py sesuai bab.

16.1 Sifat Molekuler untuk Operator Satu Partikel

Nilai harapan operator satu partikel diperoleh dari kontraksi density matrix dengan integral operator, ditambah kontribusi inti bila relevan.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 16.1 sifat molekuler untuk operator satu partikel akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

16.2 Gradien Inti Analitik

Gradien analitik menghitung turunan energi terhadap koordinat dalam satu evaluasi terstruktur, lebih efisien daripada finite difference untuk banyak koordinat.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

16.2.1 Persamaan Dasar Turunan Hartree-Fock

Turunan energi mencakup turunan integral, respons orbital, dan syarat stasioner. Pulay force muncul ketika basis bergantung pada koordinat inti.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

16.2.2 Turunan Integral

Turunan overlap, kinetik, tarik inti, dan ERI dapat dievaluasi dengan relasi integral yang diperluas.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Catat shape, dtype, order, dan kepemilikan buffer.
- Gunakan contoh kecil untuk memeriksa konvensi indeks.

16.2.3 Gradien Inti untuk Energi RHF

Gradien RHF menyusun kontribusi Hellmann-Feynman, Pulay, dan repulsi inti dalam bentuk yang dapat dikontraksikan dengan densitas.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.

- Dokumentasikan asumsi dan batas penggunaan.

16.2.4 Gradien Inti dengan Finite Difference

Finite difference mudah diterapkan tetapi memerlukan pemilihan langkah dan perhitungan banyak energi. Central difference biasanya mengurangi galat truncation.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

16.2.5 Orbital Molekul Orde Pertama

Respons orbital diperoleh dari coupled-perturbed HF. Solusi tidak selalu perlu dibentuk penuh jika sifat tertentu dapat dihitung melalui persamaan adjoint.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

16.3 Solver Persamaan Linear Subruang Krylov

GMRES, MINRES, dan varian Krylov menggunakan aksi operator dan basis subruang. Preconditioner penting untuk persamaan respons yang ill-conditioned.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

- Nyatakan bentuk input dan output, termasuk tipe, satuan, serta dimensi array.
- Pisahkan kernel numerik dari orkestrasi, pencatatan, dan penyimpanan.
- Sediakan uji kasus kecil, uji invariant, dan toleransi numerik yang terdokumentasi.
- Ukur biaya alokasi, transfer, atau komunikasi selain waktu kernel utama.

Pertanyaan desain: Bagaimana 16.3 solver persamaan linear subruang krylov akan berperilaku ketika ukuran molekul, jumlah basis, atau jumlah pekerjaan meningkat sepuluh kali?

16.4 Gradien Inti dengan Diferensiasi Otomatis

AD dapat menurunkan kode integral dan SCF, tetapi iterasi fixed-point memerlukan strategi unrolling atau implicit differentiation.

Dalam implementasi kimia kuantum, bentuk tensor, konvensi indeks, simetri, satuan, dan kriteria konvergensi merupakan bagian dari definisi algoritma. Setiap optimasi perlu divalidasi terhadap kasus kecil yang dapat dihitung dengan jalur referensi.

16.4.1 Kelas Python yang Dapat Didiferensiasi

Objek perlu memisahkan data differentiable, static configuration, dan efek samping. PyTree atau struktur serupa membantu transformasi fungsi.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

16.4.2 Diferensiasi Implisit

Fixed-point dapat didiferensiasi melalui teorema fungsi implisit tanpa menyimpan seluruh riwayat iterasi. Diperlukan solusi sistem linear adjoint.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

16.4.3 Turunan Orde Tinggi

Komposisi JVP dan VJP membentuk Hessian-vector product tanpa membentuk Hessian penuh.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

16.4.4 Konversi antara JVP dan VJP

JVP mendorong perturbasi ke depan, VJP menarik kovektor ke belakang. Pilihan tergantung rasio dimensi input-keluaran dan kebutuhan memori.

Hubungkan implementasi dengan persamaan dan indeks yang digunakan. Sertakan pemeriksaan dimensi, hermitisitas atau antisimetri, konservasi partikel, serta residual. Dengan cara ini, bug indeks dapat ditemukan sebelum menyebar ke energi atau sifat akhir.

- Tuliskan satu uji keberhasilan dan satu uji kegagalan.
- Dokumentasikan asumsi dan batas penggunaan.

Praktikum Terpandu

Menghitung gradien RHF dengan PySCF dan memvalidasi salah satu komponen menggunakan finite difference energi.

Contoh kode Bab 16

```
from pyscf import gto, scf, grad

mol = gto.M(
    atom="H 0 0 0; H 0 0 0.75",
    basis="sto-3g",
    unit="Angstrom",
    verbose=0,
)
mf = scf.RHF(mol).run(conv_tol=1e-12)
g = grad.RHF(mf).kernel()
print("gradien atom 1:", g[0])
print("gradien atom 2:", g[1])
print("jumlah gaya translasi:", g.sum(axis=0))
```

Validasi: Jalankan contoh pada sistem uji, catat versi Python dan pustaka, lalu bandingkan keluaran dengan perhitungan referensi atau invariant yang sesuai. Jangan menganggap keberhasilan eksekusi sebagai bukti kebenaran ilmiah.

Ringkasan Bab

- Sifat molekuler merupakan turunan atau nilai harapan yang menuntut konsistensi basis dan respons.
- Finite difference berguna sebagai validasi tetapi mahal dan sensitif terhadap langkah.
- Diferensiasi implisit dan Krylov menghindari penyimpanan riwayat serta matriks penuh.

Latihan dan Refleksi

46. Turunkan rumus central difference dan orde galatnya.
47. Jelaskan asal Pulay force pada basis berpusat atom.
48. Bandingkan biaya JVP dan VJP untuk fungsi energi skalar dari ribuan parameter.

Proyek mini: Menghitung gradien RHF dengan PySCF dan memvalidasi salah satu komponen menggunakan finite difference energi.

LAMPIRAN A
RESEP LINGKUNGAN KOMPUTASI

Lampiran ini menyediakan pola instalasi yang dapat disesuaikan. Nama dan versi paket berubah dari waktu ke waktu; gunakan dokumentasi resmi dan lock file proyek untuk lingkungan produksi.

Lingkungan venv dasar

```
python -m venv .venv
# Linux/macOS
source .venv/bin/activate
# Windows PowerShell
.venv\Scripts\Activate.ps1
python -m pip install --upgrade pip
python -m pip install numpy scipy pandas matplotlib pyscf sympy numba h5py
```

Contoh environment.yml

```
name: qc-python
channels:
  - conda-forge
dependencies:
  - python
  - numpy
  - scipy
  - pandas
  - matplotlib
  - h5py
  - pip
  - pip:
      - pyscf
```

Komponen	Yang perlu dicatat
Python	versi mayor/minor dan implementasi
NumPy/SciPy	versi dan backend BLAS/LAPACK
Compiler	nama, versi, dan flag utama
CPU/GPU	model, jumlah core, kemampuan instruksi
Thread	OMP_NUM_THREADS, MKL_NUM_THREADS, worker aplikasi
Data	checksum input, basis, geometri, muatan, spin
Metode	toleransi, grid, screening, initial guess

LAMPIRAN B
KONVENSI KODE DAN VALIDASI

Konvensi berikut dirancang agar kode numerik mudah ditinjau. Ia bukan aturan mutlak, tetapi titik awal yang konsisten.

- Gunakan nama yang menunjukkan ruang indeks: ao, mo, occ, vir, atom, grid.
- Tuliskan shape pada docstring untuk setiap tensor nontrivial.
- Pisahkan satuan dari nilai atau sertakan satuan pada nama variabel.
- Jangan menggunakan toleransi tanpa nama; simpan sebagai parameter konfigurasi.
- Gunakan assert untuk invariant pengembang, dan exception untuk kesalahan input pengguna.
- Bandingkan metode cepat dengan implementasi referensi pada sistem kecil.
- Catat residual, bukan hanya energi akhir.
- Hindari global state untuk thread, cache, dan backend bila dapat diteruskan eksplisit.
- Simpan provenance: versi kode, dependensi, platform, dan checksum input.
- Gunakan seed yang dicatat untuk algoritma stokastik.

Contoh validator invariant

```
def validate_symmetric(matrix, *, atol=1e-10):
    import numpy as np
    matrix = np.asarray(matrix)
    if matrix.ndim != 2 or matrix.shape[0] != matrix.shape[1]:
        raise ValueError("matrix harus persegi")
    err = np.linalg.norm(matrix - matrix.T.conj())
    if err > atol:
        raise ValueError(f"matrix tidak Hermitian; residual={err:.3e}")
    return err
```

LAMPIRAN C
GLOSARIUM RINGKAS

Istilah	Makna
ABI	Kontrak biner antara program, termasuk calling convention, layout tipe, dan nama simbol.
Backend	Implementasi yang menjalankan operasi di balik API, misalnya BLAS, FFT, CPU, atau GPU.
Broadcasting	Aturan penyelarasan bentuk array untuk operasi elemen demi elemen tanpa replikasi fisik eksplisit.
Checkpoint	Keadaan perhitungan yang disimpan agar pekerjaan dapat dilanjutkan atau diaudit.
Contiguity	Keadaan ketika elemen array tersusun berurutan sesuai order tertentu dalam memori.
Densitas elektron	Kuantitas ruang nyata yang diperoleh dari fungsi gelombang atau orbital terisi.
Fixed-point	Keadaan x yang memenuhi x = f(x); SCF mencari fixed-point densitas.
Idempotent	Operasi yang dapat diulang tanpa mengubah hasil akhir di luar efek pertama.
Kernel	Bagian komputasi inti yang melakukan operasi numerik intensif.
Lazy evaluation	Penundaan komputasi sampai nilai benar-benar dibutuhkan.
Memoization	Penyimpanan hasil fungsi untuk input tertentu agar tidak dihitung ulang.
Provenance	Rekam asal-usul data dan hasil: input, kode, versi, platform, serta parameter.
Residual	Ukuran seberapa jauh solusi memenuhi persamaan yang ditargetkan.
Screening	Pengabaian kontribusi yang dibuktikan berada di bawah ambang tertentu.
Stride	Jarak byte antar elemen ketika bergerak pada suatu sumbu array.
Throughput	Jumlah pekerjaan yang diselesaikan per satuan waktu.
Vectorization	Pemindahan operasi berulang ke kernel array atau instruksi vektor.

Istilah	Makna
Zero-copy	Pertukaran data melalui buffer bersama tanpa menyalin isi, dengan syarat layout dan lifetime kompatibel.

DAFTAR PUSTAKA PILIHAN

Sun, Q. (2025). Python for Quantum Chemistry: A Full Stack Programming Guide. Theoretical and Computational Chemistry, Vol. 23. Elsevier.

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

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

Shendruk, T. N. dkk. Dokumentasi resmi Python, NumPy, SciPy, Pandas, Matplotlib, Numba, JAX, Dask, dan PySCF sesuai versi yang digunakan.

McWeeny, R. (1992). Methods of Molecular Quantum Mechanics. Academic Press.

Lehtola, S., Blockhuys, F., & Van Alsenoy, C. (2019). An Overview of Self-Consistent Field Calculations Within Finite Basis Sets. Molecules, 25(5), 1218.

Sun, Q. et al. (2018). PySCF: the Python-based simulations of chemistry framework. WIREs Computational Molecular Science, 8, e1340.

Harris, C. R. et al. (2020). Array programming with NumPy. Nature, 585, 357–362.

Virtanen, P. et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods, 17, 261–272.

Bradbury, J. et al. (2018). JAX: composable transformations of Python+NumPy programs. Software documentation and project materials.

PENUTUP

Kimia kuantum komputasional berkembang melalui pertemuan teori, algoritma, perangkat keras, dan rekayasa perangkat lunak. Python memiliki posisi unik karena mampu menghubungkan seluruh lapisan tersebut. Kekuatan itu hanya menghasilkan ilmu yang dapat dipercaya apabila disertai pemodelan yang tepat, pengujian, validasi numerik, dokumentasi, dan keterbukaan terhadap keterbatasan.

Pembaca dianjurkan memperlakukan setiap contoh sebagai titik awal. Ubah satu komponen pada satu waktu, ukur dampaknya, verifikasi terhadap referensi, dan simpan seluruh provenance. Dengan disiplin tersebut, kode tidak hanya menjadi alat menghitung, tetapi juga bagian dari argumen ilmiah yang dapat diperiksa dan dikembangkan bersama.


+ 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

KASMUI