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 empat bagian dan enam belas bab digunakan sebagai peta pembelajaran. Bagian 2 ditambahkan sebagai praktikum Python untuk kimia kuantum yang menghubungkan model dasar, metode variasi, teori gangguan, dan validasi numerik. Seluruh uraian, contoh, latihan, tabel, dan kode praktikum ditulis secara independen.

Rujukan konseptual untuk metode variasi dan teori gangguan mencakup Ira N. Levine, Quantum Chemistry. Soal dan pembahasan sumber tidak direproduksi; praktikum baru mengembangkan tema komputasionalnya menjadi contoh Python orisinal dengan input, kode, output yang diharapkan, validasi, dan pembahasan.

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 pemrograman: 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	Bagian/Bab yang disarankan	Tujuan
Pengembang paket ilmiah	Bagian 1, Bab 1-7	Membangun proyek, mengolah data, membuat visualisasi, serta mengelola I/O dan cloud.
Praktikum kimia kuantum	Bagian 2, Praktikum 1-35	Menerjemahkan persamaan menjadi Python; memvalidasi model dasar, metode variasi, dan teori gangguan.
Optimasi dan HPC	Bagian 3, Bab 8-11	Menghubungkan bahasa, melakukan profiling, paralelisme CPU, dan akselerasi GPU.
Pengembang metode elektronik	Bagian 4, Bab 12-16	Menerapkan integral, SCF, FCI, coupled cluster, serta sifat molekuler.

Pembaca yang ingin langsung berlatih dapat memulai dari Bagian 2 setelah menguasai dasar Python dan konsep kimia kuantum. Untuk implementasi metode struktur elektronik, lanjutkan ke Bagian 4; bila kinerja menjadi kendala, gunakan teknik pada Bagian 3. Untuk pengembangan perangkat lunak yang dipakai bersama, Bab 1, 2, 6, dan 9 tetap 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 - PRAKTIKUM PYTHON UNTUK KIMIA KUANTUM

Dari model kuantum dasar menuju metode variasi dan teori gangguan

Bagian praktikum ini dikembangkan sebagai pendamping komputasional yang mandiri. Fokus utama adalah menerjemahkan persamaan kimia kuantum ke algoritma Python, memvalidasi hasil terhadap invariant atau nilai acuan, serta menghubungkan metode numerik dengan metode variasi dan teori gangguan.

Rujukan konseptual utama: Ira N. Levine, Quantum Chemistry, khususnya tema metode variasi dan teori gangguan. Soal, kode, contoh input, dan pembahasan pada bagian ini ditulis ulang secara orisinal dan tidak dimaksudkan sebagai reproduksi soal sumber.

Panduan Menggunakan Bagian Praktikum

Setiap praktikum memakai pola tetap agar pembaca dapat memisahkan model fisik, algoritma, implementasi, dan validasi. Pola ini penting karena program yang berjalan tanpa error belum tentu menghasilkan kimia yang benar.

• Baca tujuan hitungan dan rumus kerja sebelum menjalankan kode.

• Salin kode ke berkas .py atau notebook baru; jangan mengubah banyak parameter sekaligus.

• Bandingkan output dengan nilai yang dicantumkan pada bagian Output yang Diharapkan.

• Jika hasil berbeda, periksa versi pustaka, satuan, domain, jumlah basis, grid, dan toleransi solver.

• Gunakan soal pengembangan untuk menguji sensitivitas hasil terhadap parameter.

Aturan format kode: seluruh blok kode pada bagian ini ditata dengan font Calibri 12 pt. Bila perangkat tidak memiliki Calibri, pembaca dapat memakai font metric-compatible seperti Carlito tanpa mengubah struktur kode.

Kebutuhan minimal: Python 3.11 atau lebih baru, NumPy, SciPy, dan Matplotlib. Seluruh contoh inti dirancang tanpa ketergantungan pada paket kimia kuantum eksternal sehingga dapat dijalankan pada laptop standar.

Persiapan Lingkungan Komputasi

Gunakan lingkungan virtual terpisah agar versi pustaka tercatat dan dapat direproduksi.

```
python -m venv .venv
# Windows
.venv\Scripts\activate
# Linux/macOS
source .venv/bin/activate
python -m pip install --upgrade pip
pip install numpy scipy matplotlib
```

Simpan versi paket bersama hasil praktikum. Untuk laporan, catat keluaran perintah berikut:

```
python - <<'PY'
import sys, numpy, scipy, matplotlib
print(sys.version)
print("numpy", numpy.__version__)
print("scipy", scipy.__version__)
print("matplotlib", matplotlib.__version__)
PY
```

Jika praktikum digunakan dalam kelas, dosen dapat menambahkan berkas requirements.txt atau environment.yml untuk memastikan seluruh mahasiswa memakai lingkungan yang setara.

Konvensi Satuan, Toleransi, Dan Validasi

Banyak contoh memakai satuan atomik atau satuan tereduksi agar struktur persamaan terlihat jelas. Setiap kode menyebutkan konvensi input. Jangan menggabungkan angka SI dan satuan atomik di dalam satu rumus tanpa konversi eksplisit.

• Validasi energi: bandingkan dengan nilai eksak bila tersedia atau dengan hasil basis/grid yang
lebih besar.

• Validasi vektor: uji normalisasi, ortogonalitas, dan residual Hc-Ec.

• Validasi operator: uji Hermitisitas, simetri, trace, komutator, atau konservasi probabilitas.

• Validasi konvergensi: ubah satu parameter numerik seperti N, ukuran basis, domain, atau toleransi.

• Simpan angka cukup banyak dalam komputasi; pembulatan dilakukan pada tahap pelaporan.

Toleransi praktis: untuk contoh pendidikan ini, error 10^-6 hingga 10^-8 biasanya sudah memadai. Perhitungan penelitian dapat memerlukan toleransi jauh lebih ketat dan analisis error yang lebih formal.

PETA KETERKAITAN DENGAN METODE VARIASI DAN TEORI GANGGUAN

Bagian metode variasi menekankan kuosien Rayleigh, persamaan sekuler, matriks overlap, eigenvalue, orthogonalization, dan konvergensi basis. Bagian teori gangguan menekankan koreksi energi orde pertama dan kedua, degenerasi, simetri, serta gangguan bergantung waktu. Tema-tema tersebut menjadi inti Praktikum 14-35.

KelompokPraktikum	Kemampuan utama
Fondasi numerik	1-13	satuan, grid, operator, integral, nilai eigen
Metode variasi	14-26	pivoting, Rayleigh-Ritz, Hc=ESc, Lowdin, basis convergence
Teori gangguan	27-34	orde 1-2, degenerasi, Stark, time-dependent, helium
Integrasi	35	membandingkan domain validitas metode

Dengan urutan tersebut, pembaca tidak hanya memperoleh kode jadi, tetapi juga melihat mengapa algoritma yang sama - integrasi, diagonalization, optimasi - muncul berulang dalam kimia kuantum.

Template Laporan Praktikum

Gunakan format berikut untuk setiap kegiatan. Struktur yang konsisten memudahkan audit dan perbandingan antarmahasiswa.

- Judul hitungan dan tujuan.
- Persamaan atau model yang digunakan.
- Input: parameter fisik, satuan, ukuran grid/basis, toleransi.
- Kode program dan versi pustaka.
- Output numerik serta grafik yang relevan.
- Pembahasan: arti fisik, kualitas pendekatan, sumber error.
- Validasi: nilai acuan, invariant, residual, atau uji konvergensi.
- Kesimpulan singkat dan jawaban soal pengembangan.

Untuk pekerjaan kelompok, tambahkan kolom perubahan yang dilakukan dan alasan perubahan. Hal ini mencegah laporan menjadi sekadar hasil salin-tempel kode.

Daftar Praktikum

No.	Judul Hitungan	Fokus
1	Konversi Satuan Atomik dan Energi Hartree	Fondasi
2	Energi Partikel dalam Kotak dan Spektrum Transisi	Fondasi
3	Normalisasi Numerik dan Nilai Harapan	Fondasi
4	Plot Fungsi Gelombang Partikel dalam Kotak	Fondasi
5	Superposisi Dua Keadaan dan Evolusi Fase	Fondasi
6	Persamaan Schrodinger dengan Beda Hingga: Kotak Tak Hingga	Fondasi
7	Koefisien Transmisi pada Penghalang Persegi	Fondasi
8	Sumur Potensial Berhingga dengan Beda Hingga	Fondasi
9	Osilator Harmonik dari Diagonalisasi Numerik	Fondasi
10	Osilator Anharmonik Kuartik	Fondasi
11	Matriks Momentum Sudut dan Relasi Komutator	Fondasi
12	Persamaan Radial Hidrogen dengan Grid	Fondasi
13	Elemen Matriks Dipol dan Aturan Seleksi	Fondasi
14	Eliminasi Gaussian dengan Pivoting Parsial	Variasi
15	Diagonalization Matriks Hermitian dan Uji Invariant	Variasi
16	Sensitivitas Akar Polinom Karakteristik	Variasi
17	Prinsip Rayleigh-Ritz sebagai Masalah Optimasi	Variasi
18	Persamaan Nilai Eigen Umum Hc = ESc	Variasi

DAFTAR PRAKTIKUM (LANJUTAN)

No.	Judul Hitungan	Fokus
19	Ortogonalisasi Simetris Lowdin	Variasi
20	Variasi Satu Parameter untuk Sumur Berhingga	Variasi
21	Variasi Linear dengan Basis Polinomial	Variasi
22	Variasi Linear untuk Penghalang Tengah	Variasi
23	Konvergensi Basis pada Potensial Sumur Ganda	Variasi
24	Osilator Harmonik dengan Basis Partikel dalam Kotak	Variasi
25	Osilator Kuartik dengan Basis Partikel dalam Kotak	Variasi
26	Persamaan Radial Hidrogen dengan Basis PIB	Variasi
27	Koreksi Gangguan Orde Pertama pada Penghalang Tengah	Gangguan
28	Koreksi Gangguan Orde Kedua dan Simetri Paritas	Gangguan
29	Pemindaian Posisi Gangguan dalam Kotak	Gangguan
30	Teori Gangguan Terdegenerasi: Matriks 2x2	Gangguan
31	Efek Stark Linear sebagai Masalah Degenerasi	Gangguan
32	Gangguan Bergantung Waktu: Sistem Dua Tingkat	Gangguan
33	Helium: Koreksi Tolakan Elektron Orde Pertama	Gangguan
34	Helium: Variasi Muatan Inti Efektif	Gangguan
35	Proyek Integratif: Variasi versus Gangguan	Integrasi

PRAKTIKUM 1. Konversi Satuan Atomik dan Energi Hartree

Judul hitungan. Konversi Satuan Atomik dan Energi Hartree.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Satuan atomik menyederhanakan persamaan kimia kuantum dengan menjadikan beberapa konstanta dasar bernilai satu. Praktikum ini membangun kebiasaan konversi yang eksplisit agar energi dari kode dapat dibandingkan dengan eV, bilangan gelombang, dan kJ/mol.

Contoh input. Tidak ada input molekul. Program mengambil konstanta SI dari SciPy dan mengonversi 1 Hartree.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 1 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
from scipy.constants import physical_constants, e, h, c, N_A
Eh = physical_constants["Hartree energy"][0]
print("1 Hartree =", Eh/e, "eV")
print("1 Hartree =", Eh/(h*c*100), "cm^-1")
print("1 Hartree =", Eh*N_A/1000, "kJ/mol")
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 1 - OUTPUT YANG DIHARAPKAN

```
1 Hartree = 27.211386246 eV
1 Hartree = 219474.631 cm^-1
1 Hartree = 2625.500 kJ/mol
```

Pembahasan. Nilai konversi harus konsisten lintas tiga satuan. Dalam proyek kimia kuantum, simpan energi internal dalam satu satuan kanonik lalu lakukan konversi hanya pada antarmuka keluaran.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 1 - SOAL DAN PEMBAHASAN

Soal 1. Konversikan 0,125 Hartree ke eV dan kJ/mol.

Pembahasan 1. Karena konversi linear, 0,125 Hartree = 3.401423 eV dan 328.187 kJ/mol.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 2. Energi Partikel dalam Kotak dan Spektrum Transisi

Judul hitungan. Energi Partikel dalam Kotak dan Spektrum Transisi.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Model partikel dalam kotak memberikan spektrum diskret E_n yang berbanding n^2/L^2. Di sini Python dipakai untuk menghitung tingkat energi dan frekuensi transisi sehingga hubungan struktur tingkat energi dengan spektrum dapat dilihat langsung.

Contoh input. L = 0,50 nm; partikel = elektron; tingkat n = 1-5.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 2 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
from scipy.constants import hbar, h, m_e, e, c
L = 0.5e-9
n = np.arange(1, 6)
E = n**2*np.pi**2*hbar**2/(2*m_e*L**2)/e
for ni, Ei in zip(n, E):
   print(ni, Ei)
nu = (E[1]-E[0])*e/h
print("nu =", nu)
print("lambda_nm =", c/nu*1e9)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 2 - OUTPUT YANG DIHARAPKAN

```
Energi PIB (eV):
n=1: 1.504121
n=2: 6.016483
n=3: 13.537086
n=4: 24.065930
n=5: 37.603016
Transisi 1->2: nu = 1.091084e+15 Hz, lambda = 274.766 nm
```

Pembahasan. Karena E_n berbanding n^2, jarak tingkat membesar saat n naik. Panjang gelombang transisi yang diperoleh berada pada wilayah ultraviolet untuk kotak subnanometer.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 2 - SOAL DAN PEMBAHASAN

Soal 1. Ubah L dari 0,50 nm menjadi 0,60 nm. Prediksi arah perubahan seluruh energi sebelum menjalankan program.

Pembahasan 1. Energi berbanding 1/L^2. Ketika L naik dari 0,50 menjadi 0,60 nm, semua energi dikalikan (0,50/0,60)^2 = 0,69444; spektrum menjadi lebih rapat.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 3. Normalisasi Numerik dan Nilai Harapan

Judul hitungan. Normalisasi Numerik dan Nilai Harapan.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Fungsi coba yang belum ternormalisasi dapat dipakai selama pembilang dan penyebut ditangani konsisten. Integrasi numerik memperlihatkan bagaimana normalisasi, nilai harapan, dan probabilitas interval dihitung pada grid.

Contoh input. L=1 (satuan tereduksi); fungsi coba f=x(L-x); interval probabilitas 0,25L-0,75L.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 3 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
L = 1.0
x = np.linspace(0, L, 20001)
f = x*(L-x)
norm = np.sqrt(np.trapezoid(f*f, x))
psi = f/norm
mean_x = np.trapezoid(psi**2*x, x)
mask = (x >= 0.25) & (x <= 0.75)
p_mid = np.trapezoid(psi[mask]**2, x[mask])
print(1/norm, mean_x, p_mid)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 3 - OUTPUT YANG DIHARAPKAN

Konstanta normalisasi N = 5.47722558
<x> = 0.50000000 L
P(0.25L<x<0.75L) = 0.79296875

Pembahasan. Simetri fungsi coba terhadap pusat kotak memaksa <x>=L/2. Probabilitas tengah lebih besar daripada panjang intervalnya karena fungsi menumpuk di pusat.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 3 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 4. Plot Fungsi Gelombang Partikel dalam Kotak

Judul hitungan. Plot Fungsi Gelombang Partikel dalam Kotak.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Visualisasi psi dan |psi|^2 membantu menghubungkan bilangan kuantum dengan jumlah simpul, simetri, dan daerah probabilitas tinggi. Grafik juga berguna sebagai uji kewajaran sebelum perhitungan yang lebih rumit.

Contoh input. Kotak 0<=x<=L dengan L=1; tampilkan n=1,2,3 dan deteksi maksimum densitas hingga n=4.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 4 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

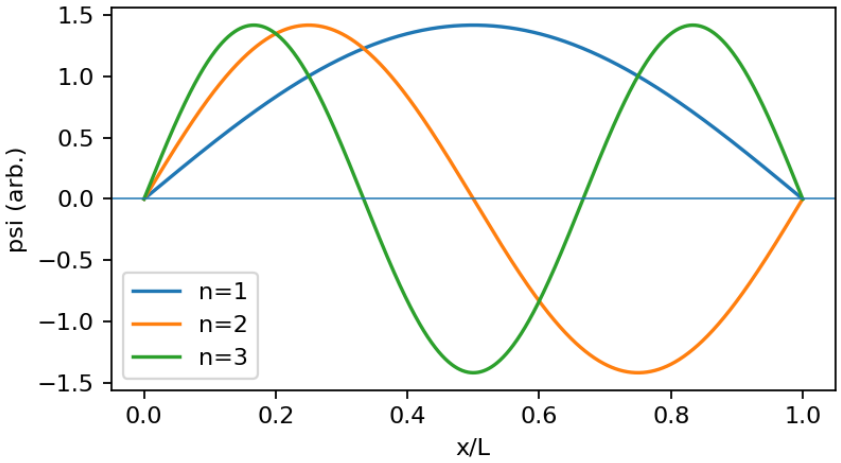
```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 1, 1000)
for n in (1, 2, 3):
   psi = np.sqrt(2)*np.sin(n*np.pi*x)
   plt.plot(x, psi, label=f"n={n}")
for n in (1,2,3,4):
   psi=np.sqrt(2)*np.sin(n*np.pi*x)
   print(n, x[np.argmax(psi**2)])
plt.axhline(0, lw=0.7)
plt.xlabel("x/L"); plt.ylabel("psi")
plt.legend(); plt.show()
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 4 - OUTPUT YANG DIHARAPKAN

Lokasi salah satu maksimum |psi|^2:
n=1: x/L ~ 0.4995
n=2: x/L ~ 0.2503
n=3: x/L ~ 0.1662
n=4: x/L ~ 0.1251



Pembahasan. Jumlah simpul internal sama dengan n-1. Visualisasi menjadi tes cepat bahwa urutan eigenstate tidak tertukar oleh solver numerik.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 4 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 5. Superposisi Dua Keadaan dan Evolusi Fase

Judul hitungan. Superposisi Dua Keadaan dan Evolusi Fase.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Keadaan nonstasioner merupakan kombinasi fungsi eigen dengan fase waktu berbeda. Walau probabilitas total tetap satu, interferensi antar komponen membuat densitas pada titik tertentu berosilasi.

Contoh input. Superposisi setara n=1 dan n=2; titik observasi x=0,30L; waktu dinyatakan sebagai fraksi periode beat.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 5 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
x0 = 0.30
E = lambda n: n*n*np.pi**2/2
psi = lambda n, x: np.sqrt(2)*np.sin(n*np.pi*x)
omega = E(2)-E(1)
T = 2*np.pi/omega
for q in (0, .25, .5, .75, 1):
   t = q*T
   amp = (psi(1,x0)*np.exp(-1j*E(1)*t)
        +psi(2,x0)*np.exp(-1j*E(2)*t))/np.sqrt(2)
   print(q, abs(amp)**2)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 5 - OUTPUT YANG DIHARAPKAN

```
Delta omega (a.u. model) = 14.80440660
t/T=0.00: rho(0.30L)=3.097859
t/T=0.25: rho(0.30L)=1.559017
t/T=0.50: rho(0.30L)=0.020175
t/T=0.75: rho(0.30L)=1.559017
t/T=1.00: rho(0.30L)=3.097859
```

Pembahasan. Densitas di satu titik berubah karena suku interferensi mengandung cos[(E2-E1)t]. Pada satu periode beat, pola kembali ke keadaan awal.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 5 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 6. Persamaan Schrodinger dengan Beda Hingga: Kotak Tak Hingga

Judul hitungan. Persamaan Schrodinger dengan Beda Hingga: Kotak Tak Hingga.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Operator kinetik dapat didiskretkan menjadi matriks tridiagonal. Diagonalization matriks tersebut menghasilkan pendekatan nilai eigen dan fungsi eigen yang konvergen ketika grid diperhalus.

Contoh input. L=1; N=250 titik interior; operator -1/2 d^2/dx^2.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 6 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
from scipy.linalg import eigh_tridiagonal
N, L = 250, 1.0
dx = L/(N+1)
diag = np.full(N, 1/dx**2)
off = np.full(N-1, -0.5/dx**2)
E = eigh_tridiagonal(diag, off,
   select="i", select_range=(0,3))[0]
for i, Ei in enumerate(E, 1):
   exact = (i*np.pi)**2/2
   print(i, Ei, exact)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 6 - OUTPUT YANG DIHARAPKAN

n	E_FD	E_eksak	galat(%)
1	4.9347378	4.9348022	-0.0013
2	19.7381781	19.7392088	-0.0052
3	44.4080018	44.4132198	-0.0117
4	78.9403443	78.9568352	-0.0209

Pembahasan. Galat tingkat rendah kecil dan meningkat untuk n lebih tinggi pada grid tetap. Memperbesar N menurunkan galat kira-kira sesuai orde diskretisasi turunan kedua.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 6 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 7. Koefisien Transmisi pada Penghalang Persegi

Judul hitungan. Koefisien Transmisi pada Penghalang Persegi.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Tunneling adalah konsekuensi langsung dari solusi gelombang di daerah terlarang klasik. Rumus transmisi penghalang persegi menjadi contoh sederhana untuk memeriksa konservasi probabilitas T+R=1.

Contoh input. Satuan tereduksi: E=1, V0=3, lebar penghalang a=1,2.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 7 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
E, V0, a = 1.0, 3.0, 1.2
kappa = np.sqrt(V0-E)
T = 1/(1 + V0**2*np.sinh(kappa*a)**2
     /(4*E*(V0-E)))
print("T =", T)
print("R =", 1-T)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 7 - OUTPUT YANG DIHARAPKAN

```
E=1.000, V0=3.000, a=1.200
T = 0.11331606
R = 0.88668394
T+R = 1.00000000
```

Pembahasan. T kecil tetapi tidak nol karena fungsi gelombang meredam secara eksponensial di dalam penghalang. Untuk a atau V0 lebih besar, T turun cepat.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 7 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 8. Sumur Potensial Berhingga dengan Beda Hingga

Judul hitungan. Sumur Potensial Berhingga dengan Beda Hingga.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Sumur berhingga dapat diselesaikan tanpa mencari akar persamaan transendental dengan menyusun Hamiltonian pada grid. Fungsi eigen terikat terlihat memiliki ekor di luar wilayah potensial nol.

Contoh input. V0=20; lebar sumur 1; domain -3 hingga 3; 500 titik interior.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 8 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

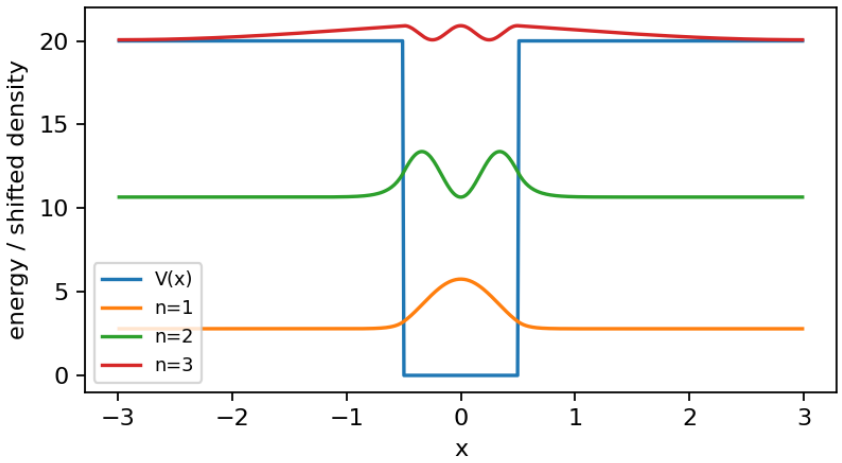
```
import numpy as np
from scipy.linalg import eigh_tridiagonal
V0, a, xmax = 20.0, 1.0, 3.0
N = 500
x = np.linspace(-xmax, xmax, N+2)[1:-1]
dx = x[1]-x[0]
V = np.where(abs(x) <= a/2, 0.0, V0)
diag = 1/dx**2 + V
off = np.full(N-1, -0.5/dx**2)
E = eigh_tridiagonal(diag, off,
   select="i", select_range=(0,3))[0]
print(E)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 8 - OUTPUT YANG DIHARAPKAN

Empat energi terikat terendah (satuan model):
E1 = 2.7884525
E2 = 10.6600816
E3 = 20.0637275
E4 = 20.7814890



Pembahasan. Keadaan terikat berada di bawah V0 dan memiliki penetrasi ke luar sumur. Domain harus cukup luas agar batas numerik tidak memotong ekor secara signifikan.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 8 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 9. Osilator Harmonik dari Diagonalisasi Numerik

Judul hitungan. Osilator Harmonik dari Diagonalisasi Numerik.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Osilator harmonik adalah benchmark utama karena energi eksaknya diketahui. Beda hingga memungkinkan verifikasi orde galat dan pengaruh ukuran domain terhadap tingkat energi rendah.

Contoh input. V=x^2/2; domain -7 hingga 7; 600 titik interior.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 9 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
from scipy.linalg import eigh_tridiagonal
N = 600
x = np.linspace(-7, 7, N+2)[1:-1]
dx = x[1]-x[0]
V = 0.5*x*x
diag = 1/dx**2 + V
off = np.full(N-1, -0.5/dx**2)
E = eigh_tridiagonal(diag, off,
   select="i", select_range=(0,4))[0]
print(E)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 9 - OUTPUT YANG DIHARAPKAN

v	E_FD	E_eksak	galat
0	0.4999830	0.5000000	-1.70e-05
1	1.4999152	1.5000000	-8.48e-05
2	2.4997795	2.5000000	-2.20e-04
3	3.4995760	3.5000000	-4.24e-04
4	4.4993046	4.5000000	-6.95e-04

Pembahasan. Energi numerik mendekati v+1/2. Penyimpangan tersisa berasal dari grid dan pemotongan domain, bukan dari model fisiknya.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 9 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 10. Osilator Anharmonik Kuartik

Judul hitungan. Osilator Anharmonik Kuartik.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Penambahan suku x^4 menghilangkan spektrum berjarak sama. Kasus ini menunjukkan manfaat solver numerik ketika persamaan Schrodinger tidak lagi memiliki solusi elementer.

Contoh input. V=x^2/2+0,05x^4; domain -7 hingga 7; 700 titik interior.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 10 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

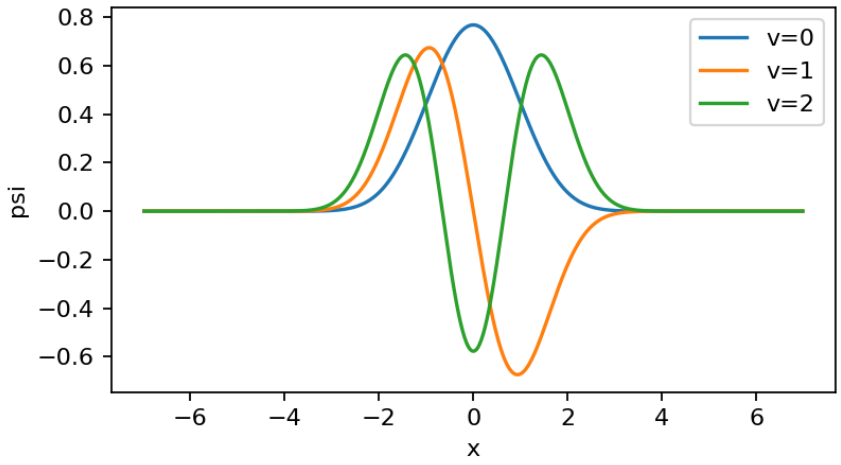
```
import numpy as np
from scipy.linalg import eigh_tridiagonal
N = 700
x = np.linspace(-7, 7, N+2)[1:-1]
dx = x[1]-x[0]
V = 0.5*x*x + 0.05*x**4
diag = 1/dx**2 + V
off = np.full(N-1, -0.5/dx**2)
E = eigh_tridiagonal(diag, off,
   select="i", select_range=(0,3))[0]
print(E)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 10 - OUTPUT YANG DIHARAPKAN

Energi V=0.5 x^2 + 0.05 x^4:
E0 = 0.53262748
E1 = 1.65335001
E2 = 2.87373044
E3 = 4.17581145



Pembahasan. Suku kuartik menaikkan energi terutama pada keadaan lebih tereksitasi karena fungsi gelombang menjelajah |x| lebih besar.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 10 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 11. Matriks Momentum Sudut dan Relasi Komutator

Judul hitungan. Matriks Momentum Sudut dan Relasi Komutator.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Representasi matriks operator Lx, Ly, dan Lz menyediakan jembatan antara aljabar operator dan aljabar linear numerik. Relasi komutator dapat diuji sebagai invariant implementasi.

Contoh input. Bilangan kuantum l=2; hbar=1; basis m=2,1,0,-1,-2.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 11 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
l = 2
m = np.arange(l, -l-1, -1, dtype=float)
d = 2*l+1
Lp = np.zeros((d,d))
for j, mj in enumerate(m):
   mp = mj+1
   hit = np.where(m == mp)[0]
   if len(hit):
      Lp[hit[0],j] = np.sqrt(l*(l+1)-mj*(mj+1))
Lm = Lp.T
Lx, Ly = (Lp+Lm)/2, (Lp-Lm)/(2j)
Lz = np.diag(m)
print(np.linalg.norm(Lx@Ly-Ly@Lx-1j*Lz))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 11 - OUTPUT YANG DIHARAPKAN

```
l=2, dimensi=5
Eigen Lz = [-2. -1.  0.  1.  2.]
||[Lx,Ly]-iLz|| = 7.162e-16
```

Pembahasan. Norma komutator mendekati nol pada presisi floating-point. Ini uji yang kuat untuk kesalahan indeks pada konstruksi operator tangga.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 11 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 12. Persamaan Radial Hidrogen dengan Grid

Judul hitungan. Persamaan Radial Hidrogen dengan Grid.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Untuk atom hidrogen, fungsi radial u(r)=rR(r) mengubah masalah tiga dimensi menjadi persamaan satu dimensi dengan syarat batas u(0)=0. Energi l=0 dapat diperoleh dari matriks tridiagonal.

Contoh input. Atom H, l=0; r_max=30 a0; N=1800; satuan atomik.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 12 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
from scipy.linalg import eigh_tridiagonal
rmax, N = 30.0, 1800
r = np.linspace(0, rmax, N+2)[1:-1]
dr = r[1]-r[0]
diag = 1/dr**2 - 1/r
off = np.full(N-1, -0.5/dr**2)
E = eigh_tridiagonal(diag, off,
   select="i", select_range=(0,2))[0]
print(E)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 12 - OUTPUT YANG DIHARAPKAN

```
Energi radial H, l=0 (Hartree):
n=1: -0.49996532  exact=-0.50000000
n=2: -0.12499783  exact=-0.12500000
n=3: -0.05542323  exact=-0.05555556
```

Pembahasan. Tiga energi pertama mendekati -1/(2n^2) Hartree. Grid rapat diperlukan dekat r=0 karena potensial Coulomb berubah cepat.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 12 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 13. Elemen Matriks Dipol dan Aturan Seleksi

Judul hitungan. Elemen Matriks Dipol dan Aturan Seleksi.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Elemen matriks operator posisi merupakan inti aturan seleksi dipol. Struktur nol-tidak nol pada matriks menampilkan peran paritas tanpa harus mengandalkan argumen kualitatif saja.

Contoh input. Kotak L=1; m,n=1..5; operator dipol proporsional x.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 13 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
x = np.linspace(0, 1, 20001)
def psi(n):
   return np.sqrt(2)*np.sin(n*np.pi*x)
M = np.zeros((5,5))
for m in range(1,6):
   for n in range(1,6):
      M[m-1,n-1] = np.trapezoid(
         psi(m)*x*psi(n), x)
print(M)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 13 - OUTPUT YANG DIHARAPKAN

```
Matriks <m|x|n>, m,n=1..5 (L=1):
[[ 0.5     -0.18013  0.     -0.01441  0.    ]
 [-0.18013  0.5     -0.19454 -0.     -0.01838]
 [ 0.     -0.19454  0.5     -0.19851  0.    ]
 [-0.01441  0.     -0.19851  0.5     -0.20014]
 [-0.     -0.01838 -0.     -0.20014  0.5    ]]
```

Pembahasan. Elemen luar diagonal hanya muncul kuat untuk pasangan dengan paritas berbeda. Struktur ini mengkode aturan seleksi dipol satu dimensi.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 13 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 14. Eliminasi Gaussian dengan Pivoting Parsial

Judul hitungan. Eliminasi Gaussian dengan Pivoting Parsial.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Bab metode variasi menekankan sistem persamaan linear dan kestabilan numerik. Pivoting parsial menghindari pembagian oleh pivot yang sangat kecil dan mengurangi amplifikasi galat pembulatan.

Contoh input. Sistem 3x3 dengan pivot awal 10^-10 untuk menunjukkan kebutuhan pertukaran baris.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 14 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
def solve_gauss(A, b):
    A, b = A.astype(float).copy(), b.astype(float).copy()
    n = len(b)
    for k in range(n-1):
        p = k + np.argmax(abs(A[k:,k]))
        A[[k,p]], b[[k,p]] = A[[p,k]], b[[p,k]]
        for i in range(k+1,n):
            f = A[i,k]/A[k,k]
            A[i,k:] -= f*A[k,k:]
            b[i] -= f*b[k]
    x = np.zeros(n)
    for i in range(n-1,-1,-1):
        x[i]=(b[i]-A[i,i+1:]@x[i+1:])/A[i,i]
    return x
A=np.array([[1e-10,1,1],[1,2,3],[2,1,1.]])
b=np.array([2.,6.,4.])
x=solve_gauss(A,b)
print("x =",x)
print("residual =",np.linalg.norm(A@x-b))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 14 - OUTPUT YANG DIHARAPKAN

```
Solusi = [1. 1. 1.]
Norma residual = 0.000e+00
Pivoting parsial: AKTIF
```

Pembahasan. Tanpa pivoting, pivot 10^-10 menyebabkan faktor eliminasi sangat besar. Pivoting parsial memilih elemen terbesar di kolom dan menjaga residual kecil.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 14 - SOAL DAN PEMBAHASAN

Soal 1. Matikan pivoting dan bandingkan residual. Mengapa solusi dapat memburuk walau algoritma formalnya sama?

Pembahasan 1. Tanpa pivoting, eliminasi membagi dengan 10^-10 pada langkah awal sehingga error floating-point diperbesar. Residual dapat meningkat tajam atau muncul overflow/kehilangan digit signifikan.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 15. Diagonalization Matriks Hermitian dan Uji Invariant

Judul hitungan. Diagonalization Matriks Hermitian dan Uji Invariant.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Hamiltonian fisik direpresentasikan oleh matriks Hermitian. Nilai eigennya real, vektor eigennya ortonormal, dan jumlah nilai eigen sama dengan trace; ketiga sifat ini cocok sebagai uji unit.

Contoh input. Matriks Hermitian kompleks 3x3 yang memiliki coupling real dan imajiner.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 15 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
A = np.array([[2,1-1j,0],
          [1+1j,3,2j],
          [0,-2j,4]], complex)
w, C = np.linalg.eigh(A)
print(w)
print("trace =", np.trace(A).real)
print("unitarity error =",
    np.linalg.norm(C.conj().T@C-np.eye(3)))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 15 - OUTPUT YANG DIHARAPKAN

```
Eigenvalue = [0.51071143 2.71083145 5.77845712]
trace(A)=9.00000000, sum(lambda)=9.00000000
||C^H C-I||=1.245e-15
```

Pembahasan. Semua eigenvalue real dan matriks eigenvektor uniter. Jika salah satu uji gagal jauh di atas toleransi, matriks atau konvensi adjoint perlu diperiksa.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 15 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 16. Sensitivitas Akar Polinom Karakteristik

Judul hitungan. Sensitivitas Akar Polinom Karakteristik.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Polinom karakteristik orde tinggi dapat sangat ill-conditioned. Gangguan amat kecil pada koefisien dapat menggeser akar secara dramatis, sehingga diagonalization matriks lebih disukai daripada membentuk polinom karakteristik secara eksplisit.

Contoh input. Polinom dengan akar awal 1..20; koefisien lambda^19 diganggu sebesar 10^-8.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 16 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

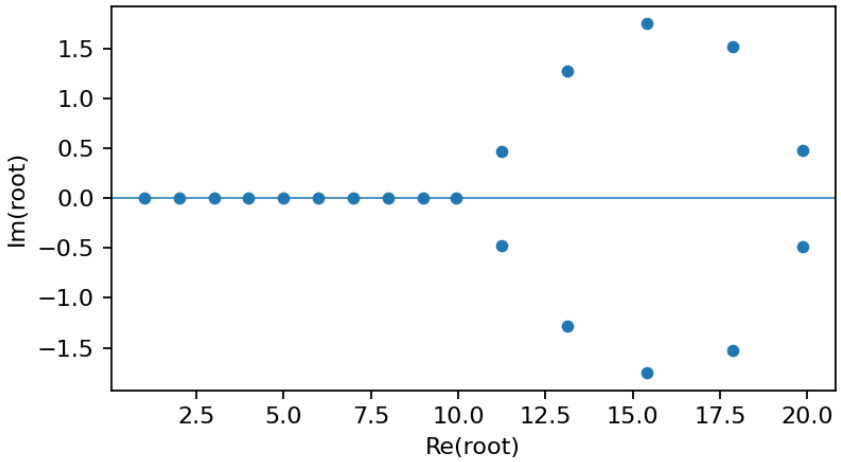
```
import numpy as np
r0 = np.arange(1,21,dtype=float)
p = np.poly(r0)
p2 = p.copy()
p2[1] += 1e-8
r = np.roots(p2)
print("real roots =",
    np.sum(abs(r.imag) < 1e-8))
print("max imaginary =",
    np.max(abs(r.imag)))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 16 - OUTPUT YANG DIHARAPKAN

Akar real setelah gangguan kecil koefisien: 10/20
Pergeseran akar maksimum terdekat ~ 1.851207
Max |Im(root)| = 1.752497



Pembahasan. Akar yang semula real dapat menjadi kompleks meskipun perubahan koefisien sangat kecil. Ini menunjukkan mengapa polinom karakteristik bukan jalur numerik yang stabil untuk eigenvalue besar.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 16 - SOAL DAN PEMBAHASAN

Soal 1. Ubah gangguan koefisien dari 10^-8 menjadi 10^-10 dan 10^-6. Bandingkan jumlah akar kompleks.

Pembahasan 1. Semakin besar gangguan koefisien, semakin besar perpindahan akar dan biasanya semakin banyak akar yang meninggalkan sumbu real. Gangguan lebih kecil mengurangi, tetapi tidak menghilangkan, sifat ill-conditioned polinom orde tinggi.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 17. Prinsip Rayleigh-Ritz sebagai Masalah Optimasi

Judul hitungan. Prinsip Rayleigh-Ritz sebagai Masalah Optimasi.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Teorema Rayleigh-Ritz menyatakan bahwa nilai eigen terendah matriks Hermitian adalah minimum kuosien Rayleigh. Praktikum ini mengubah masalah nilai eigen menjadi optimasi satu parameter.

Contoh input. Matriks H=[[2,2],[2,-1]] dan vektor ternormalisasi x=(cos theta,sin theta).

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 17 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
from scipy.optimize import minimize_scalar
H = np.array([[2.,2.],[2.,-1.]])
def rq(theta):
   x = np.array([np.cos(theta), np.sin(theta)])
   return x@H@x
res = minimize_scalar(rq, bounds=(0,np.pi),
                      method="bounded")
print(res.fun, res.x)
print(np.linalg.eigvalsh(H))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 17 - OUTPUT YANG DIHARAPKAN

```
Minimum Rayleigh quotient = -2.0000000000
theta* = 2.03444381 rad
Eigenvalue langsung = [-2.  3.]
```

Pembahasan. Minimum kuosien Rayleigh sama dengan eigenvalue terendah. Arah vektor pada theta optimum setara dengan eigenvector hingga tanda global.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 17 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 18. Persamaan Nilai Eigen Umum Hc = ESc

Judul hitungan. Persamaan Nilai Eigen Umum Hc = ESc.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Basis kimia kuantum sering tidak ortogonal sehingga persamaan sekuler berbentuk Hc=ESc. Solver generalized eigenvalue menangani matriks overlap S secara langsung dan menormalkan vektor terhadap metrik S.

Contoh input. H=[[4,1],[1,6]], S=[[2,1],[1,3]].

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 18 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
from scipy.linalg import eigh
H = np.array([[4.,1.],[1.,6.]])
S = np.array([[2.,1.],[1.,3.]])
w, C = eigh(H, S)
print(w)
print(np.diag(C.T@S@C))
print(np.linalg.norm(H@C-S@C@np.diag(w)))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 18 - OUTPUT YANG DIHARAPKAN

```
Generalized eigenvalues = [1.7101020514 2.6898979486]
S-norm eigenvectors = [1. 1.]
residual max = 1.601e-15
```

Pembahasan. Normalisasi yang relevan adalah c^TSc=1, bukan c^Tc=1. Residual Hc-ESc harus menjadi uji utama setelah solver.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 18 - SOAL DAN PEMBAHASAN

Soal 1. Verifikasi secara langsung bahwa setiap eigenvector memenuhi c^T S c = 1 dan Hc=ESc.

Pembahasan 1. Gunakan np.diag(C.T@S@C) dan residual H@c-w*S@c. Nilai pertama harus mendekati satu dan residual mendekati nol pada presisi numerik.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 19. Ortogonalisasi Simetris Lowdin

Judul hitungan. Ortogonalisasi Simetris Lowdin.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Ortogonalisasi Lowdin membentuk X=S^{-1/2}. Transformasi ini simetris dan menjaga basis baru sedekat mungkin dengan basis asal dalam pengertian kuadrat terkecil.

Contoh input. Matriks overlap 3x3 dengan overlap antarfungsi 0,35; 0,20; 0,10.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 19 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
S = np.array([[1,.35,.10],
        [.35,1,.20],
        [.10,.20,1.]])
s, U = np.linalg.eigh(S)
X = U@np.diag(1/np.sqrt(s))@U.T
print(s)
print(np.linalg.norm(X.T@S@X-np.eye(3)))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 19 - OUTPUT YANG DIHARAPKAN

```
Eigen S = [0.63353409 0.91532056 1.45114535]
||X^T S X-I|| = 2.502e-15
cond(S) = 2.29056
```

Pembahasan. Matriks X menghasilkan X^TSX=I hingga galat mesin. Nilai eigen S yang sangat kecil akan membuat 1/sqrt(s) besar dan menandakan basis hampir bergantung linear.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 19 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 20. Variasi Satu Parameter untuk Sumur Berhingga

Judul hitungan. Variasi Satu Parameter untuk Sumur Berhingga.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Metode variasi satu parameter memperbaiki fungsi coba dengan mengizinkan penetrasi ke daerah luar sumur. Minimum energi terhadap parameter harus tetap menjadi batas atas energi keadaan dasar dari Hamiltonian yang sama.

Contoh input. Sumur fisik 0..1; V0=15 di luar sumur; parameter perluasan c>0.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 20 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
from scipy.integrate import quad
from scipy.optimize import minimize_scalar
V0 = 15.0
def W(c):
   L = 1+2*c
   T = np.pi**2/L**2
   rho = lambda x: 2/L*np.sin(np.pi*(x+c)/L)**2
   P = quad(rho,-c,0)[0] + quad(rho,1,1+c)[0]
   return T + V0*P
res = minimize_scalar(W, bounds=(1e-6,1.2),
                  method="bounded")
print(res.x, res.fun, W(0))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 20 - OUTPUT YANG DIHARAPKAN

```
c optimum = 0.36615625
W_min = 4.99524584
W(c=0) = 9.86960440
```

Pembahasan. Energi minimum lebih rendah daripada fungsi c=0 karena ekor diperbolehkan menembus wilayah luar. Namun nilai variasi tetap tidak boleh di bawah energi eksak Hamiltonian.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 20 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 21. Variasi Linear dengan Basis Polinomial

Judul hitungan. Variasi Linear dengan Basis Polinomial.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Variasi linear memperluas fungsi coba menjadi kombinasi beberapa fungsi basis. Matriks H dan S kemudian membentuk masalah generalized eigenvalue; dua akar terendah memberi pendekatan untuk dua keadaan pertama.

Contoh input. Kotak L=1; basis f1=x^2(1-x), f2=x(1-x)^2.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 21 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
from scipy.linalg import eigh
x = np.linspace(0,1,40001)
f = [x**2*(1-x), x*(1-x)**2]
dx = x[1]-x[0]
df = [np.gradient(fi,dx) for fi in f]
S = np.array([[np.trapezoid(fi*fj,x) for fj in f]
        for fi in f])
H = np.array([[np.trapezoid(di*dj,x) for dj in df]
        for di in df])
w, C = eigh(H,S)
print(w)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 21 - OUTPUT YANG DIHARAPKAN

```
Variational polynomial basis:
state 1: W=9.99999998, exact=9.86960440, err=1.3212%
state 2: W=41.99999948, exact=39.47841760, err=6.3872%
```

Pembahasan. Dua kombinasi basis memisahkan simetri sekitar pusat. Energi variasional berada di atas energi eksak dan menurun ketika ruang basis diperbesar.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 21 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 22. Variasi Linear untuk Penghalang Tengah

Judul hitungan. Variasi Linear untuk Penghalang Tengah.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Penghalang lokal di tengah kotak mencampurkan basis PIB yang memiliki simetri sesuai. Menambah ukuran basis seharusnya menurunkan energi variasional menuju batas konvergen dari atas.

Contoh input. Penghalang V0=5 pada 0,25<x<0,75; basis PIB m=3,4,8,16.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 22 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
def Hbarrier(m, V0=5.0):
   x = np.linspace(0,1,8001)
   phi = np.array([np.sqrt(2)*np.sin((k+1)*np.pi*x)
               for k in range(m)])
   mask = (x>=.25)&(x<=.75)
   V = np.array([[V0*np.trapezoid(
      phi[j,mask]*phi[k,mask],x[mask])
      for k in range(m)] for j in range(m)])
   T = np.diag([((k+1)*np.pi)**2 for k in range(m)])
   return T+V
for m in (3,4,8,16):
   print(m, np.linalg.eigvalsh(Hbarrier(m))[:3])
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 22 - OUTPUT YANG DIHARAPKAN

m	W1	W2	W3
3	13.9282007	41.9784176	90.8288765
4	13.9282007	41.9404077	90.8288765
8	13.9262512	41.9391901	90.8129317
16	13.9259682	41.9389090	90.8124310

Pembahasan. Energi turun monoton atau hampir monoton saat basis ditambah. Ketidakmonotonan yang besar biasanya menunjukkan integrasi tidak cukup akurat atau basis tidak konsisten.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 22 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 23. Konvergensi Basis pada Potensial Sumur Ganda

Judul hitungan. Konvergensi Basis pada Potensial Sumur Ganda.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Potensial penghalang tinggi membentuk dua daerah berenergi rendah yang berkomunikasi melalui tunneling. Pasangan tingkat energi terendah menjadi dekat dan konvergensinya dapat dipantau terhadap jumlah fungsi basis.

Contoh input. Penghalang tinggi V0=80 pada tengah kotak; basis m=4,8,16,32.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 23 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
def Hbarrier(m, V0):
   x=np.linspace(0,1,8001)
   phi=np.array([np.sqrt(2)*np.sin((k+1)*np.pi*x)
             for k in range(m)])
   mask=(x>=.25)&(x<=.75)
   V=np.array([[V0*np.trapezoid(phi[j,mask]*phi[k,mask],
            x[mask]) for k in range(m)] for j in range(m)])
   T=np.diag([((k+1)*np.pi)**2 for k in range(m)])
   return T+V
for m in (4,8,16,32):
   print(m,np.linalg.eigvalsh(Hbarrier(m,80.0))[:4])
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 23 - OUTPUT YANG DIHARAPKAN

```
Basis convergence, central high barrier:
m= 4: 63.854075, 70.435244, 131.818498, 206.956844
m= 8: 62.241300, 70.083048, 129.481901, 204.831874
m=16: 62.102115, 70.015971, 129.445729, 204.793639
m=32: 62.086614, 70.004431, 129.444010, 204.791075
```

Pembahasan. Dua energi terendah membentuk pasangan dekat akibat tunneling antara dua sisi. Basis kecil dapat salah menggambarkan splitting karena penghalang membutuhkan representasi spasial lebih kaya.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 23 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 24. Osilator Harmonik dengan Basis Partikel dalam Kotak

Judul hitungan. Osilator Harmonik dengan Basis Partikel dalam Kotak.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Basis PIB bersifat lengkap pada interval berhingga. Dengan kotak cukup lebar, basis tersebut dapat merepresentasikan osilator harmonik dan memperlihatkan energi 1/2, 3/2, 5/2, dan seterusnya.

Contoh input. Hamiltonian -d^2/dx^2+x^2/4 pada kotak [-7,7]; basis m=8..32.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 24 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
# Basis PIB pada [-7,7], Hamiltonian -d2 + x^2/4.
def pib_H(m):
   a,b=-7.,7.
   x=np.linspace(a,b,10001); L=b-a
   phi=np.array([np.sqrt(2/L)*np.sin((k+1)*np.pi*(x-a)/L)
            for k in range(m)])
   T=np.diag([((k+1)*np.pi/L)**2 for k in range(m)])
   V=np.array([[np.trapezoid(phi[j]*(x*x/4)*phi[k],x)
            for k in range(m)] for j in range(m)])
   return T+V
for m in (8,16,24,32):
   print(m, np.linalg.eigvalsh(pib_H(m))[:5])
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 24 - OUTPUT YANG DIHARAPKAN

```
HO via basis PIB, target 0.5,1.5,...:
m= 8: 0.501268, 1.503726, 2.594562, 3.617639, 5.295226
m=16: 0.500000, 1.500000, 2.500000, 3.500002, 4.500020
m=24: 0.500000, 1.500000, 2.500000, 3.500002, 4.500019
m=32: 0.500000, 1.500000, 2.500000, 3.500002, 4.500019
```

Pembahasan. Ketika m membesar, lima energi pertama mendekati 0,5 sampai 4,5. Konvergensi ini merupakan demonstrasi konkret kelengkapan basis sinus pada interval terbatas.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 24 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 25. Osilator Kuartik dengan Basis Partikel dalam Kotak

Judul hitungan. Osilator Kuartik dengan Basis Partikel dalam Kotak.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Osilator kuartik merupakan contoh penting metode semi-komputasi pada bab variasi. Pilihan panjang kotak dan jumlah basis saling berkompromi: kotak terlalu pendek memotong ekor, terlalu panjang memperlambat konvergensi basis.

Contoh input. Potensial x^4/4 pada kotak [-5,5]; basis m=12,20,30.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 25 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
def quartic_H(m):
   a,b=-5.0,5.0; x=np.linspace(a,b,10001); L=b-a
   phi=np.array([np.sqrt(2/L)*np.sin((k+1)*np.pi*(x-a)/L)
            for k in range(m)])
   T=np.diag([((k+1)*np.pi/L)**2 for k in range(m)])
   V=np.array([[np.trapezoid(phi[j]*(x**4/4)*phi[k],x)
           for k in range(m)] for j in range(m)])
   return T+V
for m in (12,20,30):
   print(m,np.linalg.eigvalsh(quartic_H(m))[:3])
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 25 - OUTPUT YANG DIHARAPKAN

```
Quartic oscillator V=x^4/4:
m=12: E0=0.6680068, E1=2.3938714, E2=4.7010010
m=20: E0=0.6679863, E1=2.3936440, E2=4.6967954
m=30: E0=0.6679863, E1=2.3936440, E2=4.6967954
```

Pembahasan. Perubahan m memperlihatkan konvergensi dari atas. Pengujian kotak lain diperlukan untuk memisahkan galat basis dari galat pemotongan domain.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 25 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 26. Persamaan Radial Hidrogen dengan Basis PIB

Judul hitungan. Persamaan Radial Hidrogen dengan Basis PIB.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Persamaan radial Coulomb dapat didekati dengan basis sinus yang memenuhi $u(0)=u(L)=0$. Konvergensi tingkat n=1,2,3 menunjukkan bagaimana basis sederhana dapat mendekati fungsi eksponensial.

Contoh input. Coulomb -1/r; L=30; 36 fungsi basis sinus; satuan atomik.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Bangun matriks H dan, bila diperlukan, matriks overlap S; selesaikan masalah nilai eigen secara
stabil.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 26 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
L, m = 30.0, 36
r = np.linspace(1e-6,L,30001)
phi = np.array([np.sqrt(2/L)*np.sin((k+1)*np.pi*r/L)
          for k in range(m)])
T = np.diag([0.5*((k+1)*np.pi/L)**2 for k in range(m)])
V = np.array([[np.trapezoid(phi[j]*(-1/r)*phi[k],r)
          for k in range(m)] for j in range(m)])
print(np.linalg.eigvalsh(T+V)[:3])
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 26 - OUTPUT YANG DIHARAPKAN

```
Hydrogen l=0 via basis PIB (Hartree):
n=1: -0.4808073 (exact -0.5000000)
n=2: -0.1224678 (exact -0.1250000)
n=3: -0.0546459 (exact -0.0555556)
```

Pembahasan. Basis sinus yang cukup besar mereproduksi tingkat Coulomb rendah. Keadaan n lebih tinggi memerlukan domain lebih panjang karena radius karakteristik membesar.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 26 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 27. Koreksi Gangguan Orde Pertama pada Penghalang Tengah

Judul hitungan. Koreksi Gangguan Orde Pertama pada Penghalang Tengah.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Dalam teori gangguan tak-terdegenerasi, koreksi energi orde pertama adalah nilai harapan H' terhadap keadaan tak-terganggu. Penghalang tengah memberi integral yang mudah diverifikasi numerik maupun analitik.

Contoh input. Penghalang tengah V0=0,8; n=1..4; satuan dengan hbar^2/(2mL^2)=1.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Evaluasi elemen matriks gangguan dan gunakan rumus orde yang sesuai atau diagonalization
subruang degenerat.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 27 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
V0 = 0.8
x = np.linspace(0,1,20001)
mask=(x>=.25)&(x<=.75)
for n in (1,2,3,4):
   psi=np.sqrt(2)*np.sin(n*np.pi*x)
   E0=(n*np.pi)**2
   E1=V0*np.trapezoid(psi[mask]**2,x[mask])
   print(n,E0,E1,E0+E1)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 27 - OUTPUT YANG DIHARAPKAN

```
 n   E0       E1       E0+E1
 1  9.869604  0.654648  10.524252
 2  39.478418  0.400000  39.878418
 3  88.826440  0.315117  89.141557
 4  157.913670  0.400000  158.313670
```

Pembahasan. Koreksi n=1 lebih besar karena densitas keadaan dasar maksimum di tengah, tepat di lokasi penghalang. Ini adalah interpretasi geometris dari <n|H'|n>.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 27 - SOAL DAN PEMBAHASAN

Soal 1. Hitung koreksi orde pertama untuk penghalang yang dipindah ke $0<x<0,25$ dan jelaskan perubahan untuk n=1.

Pembahasan 1. Penghalang di tepi bertumpang tindih lebih kecil dengan densitas keadaan dasar daripada penghalang di tengah, sehingga koreksi orde pertama turun.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 28. Koreksi Gangguan Orde Kedua dan Simetri Paritas

Judul hitungan. Koreksi Gangguan Orde Kedua dan Simetri Paritas.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Koreksi orde kedua menjumlahkan kontribusi semua keadaan lain. Simetri paritas membuat banyak elemen matriks tepat nol, sehingga penghitungan lebih efisien dan interpretasinya lebih jelas.

Contoh input. Kasus praktikum sebelumnya; n=1; jumlah koreksi hingga m=100.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Evaluasi elemen matriks gangguan dan gunakan rumus orde yang sesuai atau diagonalization
subruang degenerat.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 28 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
V0=0.8
x=np.linspace(0,1,30001)
mask=(x>=.25)&(x<=.75)
psi=lambda n: np.sqrt(2)*np.sin(n*np.pi*x)
E0=np.pi**2
E1=V0*np.trapezoid(psi(1)[mask]**2,x[mask])
E2=0.0
for m in range(2,101):
   Hm=V0*np.trapezoid(psi(m)[mask]*psi(1)[mask],x[mask])
   E2 += Hm**2/(E0-(m*np.pi)**2)
print(E0,E1,E2,E0+E1+E2)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 28 - OUTPUT YANG DIHARAPKAN

```
E0=9.869604401
E1=0.654647908
E2(sum m<=100)=-0.000874820
E(0+1+2)=10.523377489
Variational benchmark=10.523373223
Dominant m: 3(-8.21e-04), 5(-3.04e-05), 7(-1.52e-05), 9(-3.29e-06), 11(-2.19e-06), 13(-7.98e-07)
```

Pembahasan. Koreksi orde kedua negatif untuk keadaan dasar karena semua penyebut E1-E_m bernilai negatif. Hanya keadaan dengan simetri sesuai memberikan kontribusi bukan nol.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 28 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 29. Pemindaian Posisi Gangguan dalam Kotak

Judul hitungan. Pemindaian Posisi Gangguan dalam Kotak.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Menggeser lokasi gangguan mengubah tumpang tindih antara H' dan densitas keadaan dasar. Pemindaian parameter menghasilkan kurva yang langsung memperlihatkan hubungan geometri gangguan dan koreksi energi.

Contoh input. V0=1; lebar gangguan 0,5L; pergeseran c=-0,20..0,20.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Evaluasi elemen matriks gangguan dan gunakan rumus orde yang sesuai atau diagonalization
subruang degenerat.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 29 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

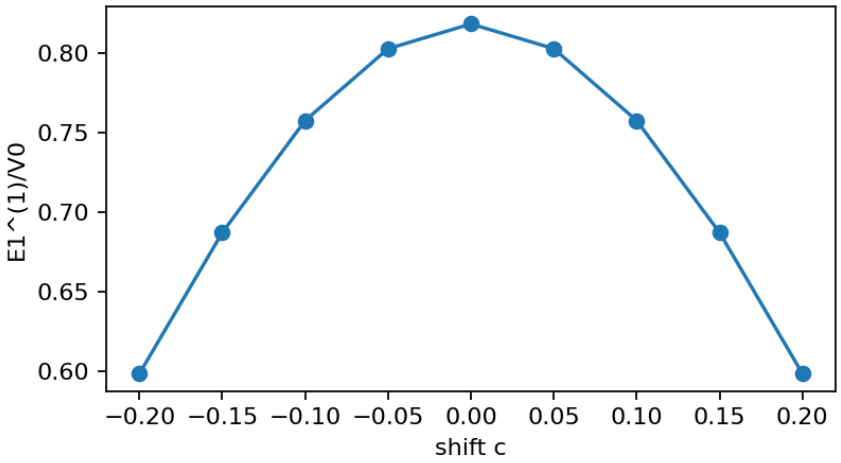
```
import numpy as np
import matplotlib.pyplot as plt
V0=1.0
x=np.linspace(0,1,25001)
psi=np.sqrt(2)*np.sin(np.pi*x)
cs=np.linspace(-.20,.20,9)
y=[]
for c in cs:
   mask=(x>=.25+c)&(x<=.75+c)
   val=V0*np.trapezoid(psi[mask]**2,x[mask])
   y.append(val); print(c,val)
plt.plot(cs,y,"o-")
plt.xlabel("shift c")
plt.ylabel("E1^(1)/V0")
plt.show()
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 29 - OUTPUT YANG DIHARAPKAN

c	E1^(1)/V0
-0.200	0.59836316
-0.150	0.68702549
-0.100	0.75751811
-0.050	0.80267832
+0.000	0.81826988
+0.050	0.80273069
+0.100	0.75751811
+0.150	0.68709786
+0.200	0.59836120



Pembahasan. Kurva simetris terhadap c=0 untuk keadaan dasar karena densitas simetris terhadap pusat. Nilai maksimum terjadi ketika gangguan menutupi daerah densitas terbesar.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 29 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 30. Teori Gangguan Terdegenerasi: Matriks 2x2

Judul hitungan. Teori Gangguan Terdegenerasi: Matriks 2x2.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Pada tingkat terdegenerasi, basis awal harus dikombinasikan agar mendiagonalkan matriks gangguan di subruang degenerasi. Nilai eigen matriks kecil tersebut adalah koreksi energi orde pertama.

Contoh input. H' = b[[4,2],[2,6]] dengan b=1 untuk angka contoh.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Evaluasi elemen matriks gangguan dan gunakan rumus orde yang sesuai atau diagonalization
subruang degenerat.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 30 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
b=1.0
Hp=b*np.array([[4.,2.],[2.,6.]])
w,C=np.linalg.eigh(Hp)
print(w)
print(C)
print("sum =",w.sum(),"trace =",np.trace(Hp))
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 30 - OUTPUT YANG DIHARAPKAN

```
Koreksi energi = [2.763932023 7.236067977] * b
Eigenvector kolom:
[[-0.8506508  0.5257311]
 [ 0.5257311  0.8506508]]
Jumlah koreksi=10.00000000 = trace=10.00000000
```

Pembahasan. Jumlah dua koreksi sama dengan trace matriks gangguan. Eigenvector memberi kombinasi basis yang benar untuk dipakai sebagai keadaan orde nol setelah gangguan dinyalakan.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 30 - SOAL DAN PEMBAHASAN

Soal 1. Buktikan secara numerik bahwa jumlah koreksi energi sama dengan trace matriks gangguan untuk tiga matriks acak Hermitian 2x2.

Pembahasan 1. Untuk matriks Hermitian mana pun, jumlah eigenvalue sama dengan trace. Karena eigenvalue H' di subruang degenerat adalah koreksi orde pertama, identitas ini berlaku langsung.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 31. Efek Stark Linear sebagai Masalah Degenerasi

Judul hitungan. Efek Stark Linear sebagai Masalah Degenerasi.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Efek Stark linear pada manifold terdegenerasi dapat dilihat sebagai diagonalization blok coupling. Contoh matriks minimal menekankan bahwa degenerasi dapat terangkat walaupun semua elemen diagonal gangguan nol.

Contoh input. Basis empat keadaan n=2; coupling 2s-2p0 = -3q; q=0,002 Hartree.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Evaluasi elemen matriks gangguan dan gunakan rumus orde yang sesuai atau diagonalization
subruang degenerat.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 31 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
q=0.002
Hp=np.zeros((4,4))
Hp[0,1]=Hp[1,0]=-3*q
w,C=np.linalg.eigh(Hp)
print(w)
print("splitting =",w[-1]-w[0])
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 31 - OUTPUT YANG DIHARAPKAN

```
Skala q=e a0 E=0.002000 Hartree
Koreksi Stark = [-0.006  0.     0.     0.006]
Pecahan = 0.01200000 Hartree
```

Pembahasan. Dua keadaan terpecah simetris menjadi +/-3q sedangkan dua lainnya tetap nol pada model blok. Ini adalah pola dasar efek Stark linear dalam subruang terdegenerasi.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 31 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 32. Gangguan Bergantung Waktu: Sistem Dua Tingkat

Judul hitungan. Gangguan Bergantung Waktu: Sistem Dua Tingkat.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Gangguan bergantung waktu menimbulkan transisi. Model dua tingkat dalam pendekatan rotating-wave menghasilkan osilasi Rabi dan menunjukkan bahwa detuning menurunkan probabilitas eksitasi maksimum.

Contoh input. Frekuensi Rabi Omega=0,08; detuning 0; 0,04; 0,12; waktu 0..250.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Evaluasi elemen matriks gangguan dan gunakan rumus orde yang sesuai atau diagonalization
subruang degenerat.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 32 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

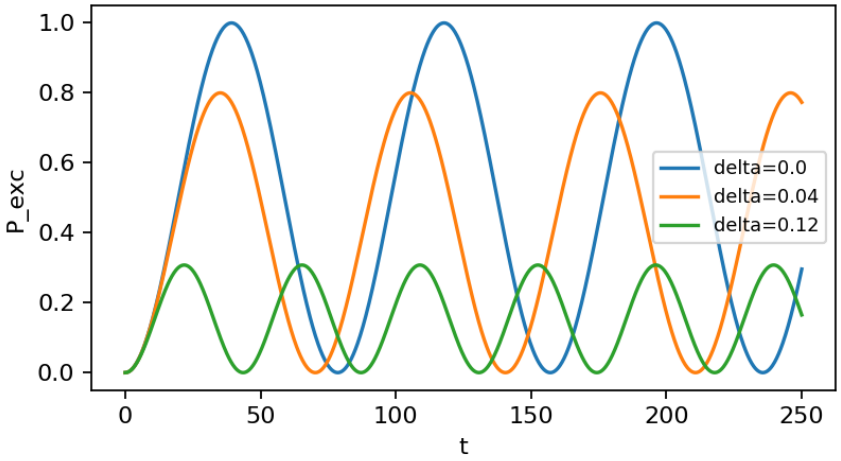
```
import numpy as np
import matplotlib.pyplot as plt
Omega=0.08
t=np.linspace(0,250,1000)
for delta in (0.0,0.04,0.12):
    Om=np.sqrt(Omega**2+delta**2)
    P=Omega**2/Om**2*np.sin(Om*t/2)**2
    print(delta,P.max())
    plt.plot(t,P,label=f"delta={delta}")
plt.legend(); plt.show()
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 32 - OUTPUT YANG DIHARAPKAN

Maksimum probabilitas eksitasi:
delta=0.000: Pmax=0.999999
delta=0.040: Pmax=0.800000
delta=0.120: Pmax=0.307692



Pembahasan. Pada resonansi, P_exc dapat mencapai hampir satu. Detuning membatasi amplitudo maksimum dengan faktor Omega^2/(Omega^2+delta^2).

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 32 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 33. Helium: Koreksi Tolakan Elektron Orde Pertama

Judul hitungan. Helium: Koreksi Tolakan Elektron Orde Pertama.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Untuk helium, Hamiltonian nol dapat dipilih sebagai jumlah dua Hamiltonian mirip hidrogen dan tolakan elektron-elektron diperlakukan sebagai gangguan. Koreksi orde pertama memberikan perbaikan besar tetapi masih menyisakan korelasi elektron.

Contoh input. Z=2; energi Hartree dikonversi ke eV; koreksi orde tinggi dipakai sebagai ilustrasi pembandingan.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Evaluasi elemen matriks gangguan dan gunakan rumus orde yang sesuai atau diagonalization
subruang degenerat.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 33 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
from scipy.constants import physical_constants, e
Eh=physical_constants["Hartree energy"][0]/e
Z=2
E0=-Z**2*Eh
E1=(5*Z/8)*Eh
E2,E3=-4.29,0.12
print(E0,E1,E0+E1)
print(E0+E1+E2+E3)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 33 - OUTPUT YANG DIHARAPKAN

```
E0 = -108.8455 eV
E1 = +34.0142 eV
E0+E1 = -74.8313 eV
Tambahan ilustratif E2=-4.29 eV, E3=+0.12 eV
Total s.d. orde 3 = -79.0013 eV
```

Pembahasan. Koreksi orde pertama mengurangi galat besar model dua elektron bebas-interaksi. Koreksi berikutnya berhubungan dengan pencampuran konfigurasi dan korelasi elektron.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 33 - SOAL DAN PEMBAHASAN

Soal 1. Ulangi hitungan dengan mengubah satu parameter utama sebesar 20%. Jelaskan perubahan hasil dari sudut pandang model fisik, bukan hanya menyebut angka.

Pembahasan 1. Jawaban harus menunjukkan ketergantungan numerik yang konsisten dengan persamaan. Untuk parameter yang memperkuat potensial atau mempersempit skala panjang, energi biasanya berubah mengikuti skala operator terkait. Laporkan angka baru dan kaitkan dengan suku Hamiltonian yang berubah.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 34. Helium: Variasi Muatan Inti Efektif

Judul hitungan. Helium: Variasi Muatan Inti Efektif.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Metode variasi helium memperkenalkan muatan inti efektif zeta. Minimisasi energi memberi screening sederhana dan energi yang lebih rendah daripada hasil gangguan orde pertama, sesuai sifat batas atas metode variasi.

Contoh input. Z=2; fungsi energi E(zeta)=zeta^2-2Zzeta+(5/8)zeta dalam Hartree.

Alur perhitungan.

• Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.

• Evaluasi elemen matriks gangguan dan gunakan rumus orde yang sesuai atau diagonalization
subruang degenerat.

• Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.

• Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 34 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

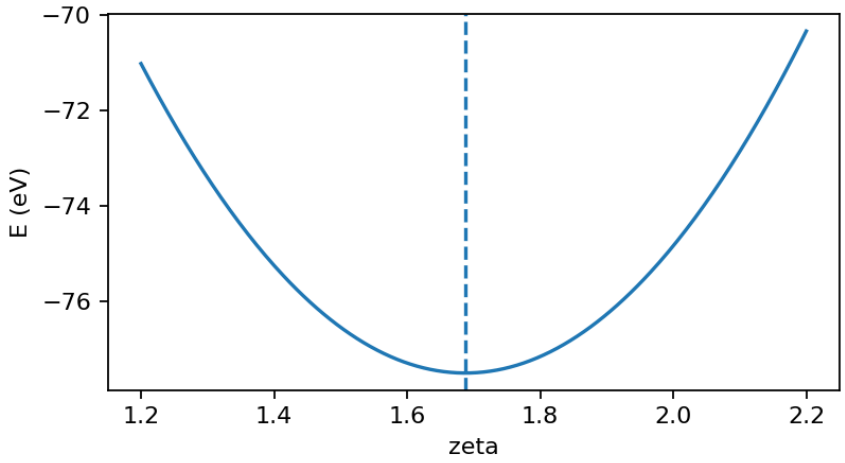
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import physical_constants, e
Eh=physical_constants["Hartree energy"][0]/e
Z=2.0
def E(z):
   return (z*z-2*Z*z+5*z/8)*Eh
zeta=Z-5/16
z=np.linspace(1.2,2.2,300)
plt.plot(z,[E(x) for x in z])
plt.axvline(zeta,ls="--")
print(zeta,E(zeta))
plt.show()
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 34 - OUTPUT YANG DIHARAPKAN

zeta analitik = 1.68750000
zeta grid minimum ~ 1.68829431
E_var(zeta*) = -77.48867 eV
Muatan screening = 0.31250



Pembahasan. Minimum muncul pada zeta=Z-5/16. Hasil variasi lebih rendah daripada pendekatan gangguan orde pertama namun tetap lebih tinggi daripada energi eksak, sesuai teorema variasi.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 34 - SOAL DAN PEMBAHASAN

Soal 1. Hitung E(zeta) pada zeta=2 dan zeta=27/16, lalu jelaskan mengapa nilai optimum lebih rendah.

Pembahasan 1. zeta=27/16 muncul dari dE/dzeta=0. Energi pada zeta=2 belum memasukkan screening optimal; penurunan zeta menyeimbangkan energi kinetik, tarikan inti, dan tolakan elektron.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

PRAKTIKUM 35. Proyek Integratif: Variasi versus Gangguan

Judul hitungan. Proyek Integratif: Variasi versus Gangguan.

Tujuan. Mengimplementasikan model secara numerik, menghasilkan output terverifikasi, dan mengidentifikasi batas pendekatan atau sumber error yang paling penting.

Dasar teori. Praktikum integratif membandingkan hasil gangguan orde pertama dengan diagonalization basis yang bersifat variasional pada keluarga penghalang yang sama. Perbedaan keduanya menjadi indikator praktis kapan gangguan mulai terlalu kuat.

Contoh input. Penghalang tengah dengan V0=0,2;0,5;1;2;5 dan basis variasional m=40.

Alur perhitungan.

- Nyatakan Hamiltonian, fungsi basis, atau rumus kerja dalam satuan yang konsisten.
- Bangun array/grid/matriks dan evaluasi kuantitas numerik yang diperlukan.
- Gunakan solver NumPy/SciPy yang sesuai dan simpan hasil dengan presisi penuh.
- Uji paling sedikit satu invariant atau nilai acuan sebelum melakukan interpretasi fisik.

PRAKTIKUM 35 - KODE PROGRAM

Kode berikut dibuat ringkas agar hubungan antara persamaan dan operasi numerik mudah ditelusuri. Untuk proyek nyata, tambahkan fungsi, validasi input, pengujian, logging, dan penyimpanan provenance.

```
import numpy as np
x=np.linspace(0,1,8001); mask=(x>=.25)&(x<=.75)
psi=np.sqrt(2)*np.sin(np.pi*x); E0=np.pi**2
def Hbarrier(m,V0):
   phi=np.array([np.sqrt(2)*np.sin((k+1)*np.pi*x)
            for k in range(m)])
   V=np.array([[V0*np.trapezoid(phi[j,mask]*phi[k,mask],
           x[mask]) for k in range(m)] for j in range(m)])
   T=np.diag([((k+1)*np.pi)**2 for k in range(m)])
   return T+V
for V0 in (0.2,0.5,1.0,2.0,5.0):
   Epert=E0+V0*np.trapezoid(psi[mask]**2,x[mask])
   Evar=np.linalg.eigvalsh(Hbarrier(40,V0))[0]
   print(V0,Epert,Evar,Epert-Evar)
```

Penjelasan kode.

Urutan operasi mengikuti alur teori: parameter didefinisikan terlebih dahulu, kemudian objek numerik dibentuk, solver dijalankan, dan keluaran diverifikasi. Hindari menyembunyikan satuan di dalam konstanta misterius. Bila matriks dibentuk, periksa shape dan simetri sebelum diagonalization; bila integral dihitung pada grid, lakukan uji konvergensi dengan menambah jumlah titik.

PRAKTIKUM 35 - OUTPUT YANG DIHARAPKAN

V0	E_pert(1)	E_variational	selisih
0.2	10.03326638	10.03321164	+5.474e-05
0.5	10.27875934	10.27841660	+3.427e-04
1.0	10.68791427	10.68653914	+1.375e-03
2.0	11.50622414	11.50069009	+5.534e-03
5.0	13.96115375	13.92592839	+3.523e-02

Pembahasan. Untuk V0 kecil, hasil gangguan dan diagonalization hampir berimpit. Saat V0 membesar, selisih meningkat sehingga orde pertama tidak lagi cukup sebagai pendekatan kuantitatif.

Perbedaan pada digit terakhir masih dapat terjadi karena versi BLAS/LAPACK, compiler, atau metode quadrature. Yang penting adalah tren, invariant, dan tingkat presisi yang dibutuhkan untuk tujuan ilmiah. Bila output menyimpang besar, jangan langsung menaikkan toleransi; periksa terlebih dahulu model, satuan, syarat batas, dan definisi matriks.

PRAKTIKUM 35 - SOAL DAN PEMBAHASAN

Soal 1. Tentukan nilai V0 terkecil dari tabel ketika selisih metode gangguan dan variasi melebihi 10^-2 pada satuan model.

Pembahasan 1. Baca kolom selisih dan pilih V0 pertama dengan |selisih|>10^-2. Nilai tersebut menandai awal ketidakcukupan orde pertama untuk toleransi yang dipilih.

Soal 2. Lakukan satu uji konvergensi: tambah ukuran grid/basis atau ketatkan toleransi. Laporkan selisih hasil dan tentukan apakah perhitungan awal sudah memadai.

Pembahasan 2. Jalankan sedikitnya tiga ukuran grid/basis atau tiga toleransi. Jika selisih antar dua hasil terakhir jauh lebih kecil daripada ketelitian pelaporan, parameter awal dapat dianggap cukup. Bila tidak, tingkatkan resolusi sampai tren konvergen terlihat. Jangan menyimpulkan konvergensi hanya dari satu pasangan data.

Checklist validasi: satuan konsisten; output dapat direproduksi; invariant lulus; parameter numerik dicatat; interpretasi membedakan error model dari error numerik.

BAGIAN 3 - 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 4 - 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 $\exp(T)|\Phi 0\rangle$ 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.

Zero-copy Pertukaran data melalui buffer bersama tanpa menyalin isi, dengan syarat layout dan lifetime kompatibel.

DAFTAR PUSTAKA PILIHAN

Levine, I. N. Quantum Chemistry, 7th ed. Pearson. Rujukan konseptual untuk metode variasi dan teori gangguan pada Bagian 2.

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., Jorgensen, P., & Olsen, J. (2000). Molecular Electronic-Structure Theory. Wiley.

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.

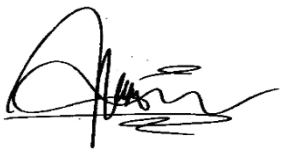
Bradbury, J. et al. (2018). JAX: composable transformations of Python+NumPy programs. Software documentation and project materials.

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.


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