LEMBAGA ILMU PENGETAHUAN INDONESIA
LIPI
INDONESIAN INSTITUTE OF SCIENCES

Quasi.id

MODUL PRAKTIS

"Research Starter Kit"

untuk riset **Teori dan Komputasi Material**

Disusun oleh:

Sasfan Arman Wella

Didukung oleh:

Edi Suprayoga

Ahmad Ridwan Tresna Nugraha

Eddwi Hesky Hasdeo

M. Shoufie Ukhtary

Muhammad Yusrul Hanna

QUAntum design and **SI**mulation Laboratory

Pusat Penelitian Fisika

Lembaga Ilmu Pengetahuan Indonesia

2021

v1.0 - Juli 2021

Prakata

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, kami dapat menyelesaikan penulisan Modul Praktis "*Research Starter Kit*" yang secara khusus diperuntukkan sebagai bahan persiapan penelitian teori dan komputasi material. Namun, kami juga berharap, modul ini bisa juga berguna untuk kepentingan yang lebih luas, sehingga dapat membantu meningkatkan kualitas sumber daya manusia Indonesia melalui aktivitas ilmiah. Kami juga ingin berterima kasih kepada semua pihak yang telah memberikan bantuan dan saran dalam penyusunan modul ini.

Kami juga menyadari masih banyak kekurangan dalam penyusunan modul ini, untuk itu kami mengharapkan saran dan masukan untuk penyempurnaan modul ini. Semoga modul ini dapat dimanfaatkan sebaik-baiknya baik oleh penulis maupun oleh pembaca.

Tangerang Selatan, Juli 2021

Tim Quasi

Sasfan A. Wella
sasf001@lipi.go.id

Edi Suprayoga
edis008@lipi.go.id

Ahmad R. T. Nugraha
ahma079@lipi.go.id

Eddwi H. Hasdeo
eddw001@lipi.go.id

M. Shoufie Ukhtary
msho001@lipi.go.id

M. Yusrul Hanna
muha210@lipi.go.id

Daftar Isi

Prakata **2**

Daftar Isi **3**

1 *Density Functional Theory* **8**

1.1 Pemodelan Material dengan Komputasi Prinsip Pertama 8

1.2 Persamaan Schrödinger Banyak Partikel 10

1.2.1 Persamaan Schrödinger untuk Atom Hidrogen 10

1.2.2 Persamaan Schrödinger Banyak Partikel 11

1.3 Density Functional Theory 16

1.3.1 Fungsional dari Densitas 16

1.3.2 Perhitungan *self-consistent field* (SCF) 18

2 Quantum Espresso **20**

2.1 Instalasi Quantum Espresso 20

2.2 Input Data Quantum Espresso 22

2.3 *Pseudopotential* 27

2.4 Parameter Konvergensi 28

2.4.1 Menentukan nilai `ecutwfc` 28

2.4.2 Menentukan nilai `ecutrho` 30
2.4.3 Menentukan nilai `k-points` 30
2.5 Struktur Setimbang 32
2.5.1 Struktur Setimbang dari Molekul Diatomik 32
2.5.2 Energi Ikat dari Molekul Diatomik 34
2.5.3 Struktur Setimbang dari Kristal 3-Dimensi 36
2.5.4 Energi Kohesi dari Kristal 3-Dimensi 37
2.6 Visualisasi Struktur Kristal 39
2.6.1 XCrySDen 39
2.6.2 VESTA 42
2.7 Relaksasi Struktur 45
2.7.1 Optimasi Koordinat Atom 45
2.7.2 Optimasi Koordinat Atom dan Sel Unit 47
2.8 Menghitung *Density of States* dan *Band Gap* 49
2.8.1 Perhitungan `scf` 49
2.8.2 Perhitungan `nscf` 50
2.8.3 Perhitungan `dos` 51
2.8.4 Membuat Grafik `dos` 52
2.9 Menghitung *Band Structure* dan Visualisasi *Wavefunctions* 53
2.9.1 *Band Structure* Kristal Silikon 53
2.9.2 Visualisasi *Wavefunctions* Kristal Silikon 57
2.9.3 *Band Structure* Kristal Tembaga 59

3 BoltzTraP2 64

3.1 Instalasi BoltzTrap2 64

3.2 Menghitung Koefisien Transpor Elektronik GaAs 65
3.2.1 Menghitung DOS dengan QUANTUM ESPRESSO 66
3.2.2 Memperoleh Nilai Energi Fermi 68
3.2.3 Mengkonversi Format *File* Output QE 68
3.2.4 Melakukan Interpolasi Pita Energi 69
3.2.5 Melakukan Integrasi Hasil Interpolasi 69
3.2.6 Visualisasi Koefisien Seebeck 70

4 Wannier90 72

4.1 Instalasi WANNIER90 72
4.2 Konstruksi Hamiltonian Model *Tight-binding* pada *Graphene* 73

5 Gnuplot dan Visualisasi Data 84

5.1 Instalasi GNUPLOT 85
5.2 Penggunaan Dasar GNUPLOT 86
5.3 Impor, Modifikasi, dan Ekspor Gambar dengan Inkscape 90
5.4 Variasi Bentuk Data 2D 92
5.5 Penataan Sumbu, Legenda, dan Judul dari suatu Grafik 94
5.6 Visualisasi Data dengan *Error Bar*, *Fitting*, dan Fungsi Matematis 96

6 *Symbolic* Python 99

6.1 Limit, Turunan dan Integral 100
6.1.1 Limit 100
6.1.2 Turunan 101
6.1.3 Integral 101
6.2 Ekspansi fungsi, Akar dari fungsi, dan Matriks 103

6.2.1 Ekspansi fungsi 103
6.2.2 Akar dari fungsi 103
6.2.3 Matriks 104
6.3 Nilai numerik dari ekspresi analitik 105

7 Menulis Naskah dengan LaTeX 107

7.1 Instalasi LaTeX 107
7.1.1 Mode *Offline* 107
7.1.2 Mode *Online* 109
7.2 *Template* Standar LaTeX 110
7.3 Manipulasi teks (sederhana) 111
7.3.1 Mengubah Ukuran Huruf 111
7.3.2 Mengubah *Style* Huruf 112
7.3.3 Penekanan Kata / *Emphasis* 113
7.3.4 Jenis Huruf 114
7.3.5 Penjajaran Teks / *Alignment* 114
7.3.6 Indentasi 116
7.4 Notasi dan Formula Matematis 117
7.4.1 Tampilan Formula Matematis pada LaTeX 118
7.4.2 Notasi Dasar 120
7.4.3 Label pada Formula Matematis 127
7.5 Membuat Tabel 128
7.5.1 Perbedaan Blok `Tabular` dan `Table` 128
7.5.2 Penggabungan Sel (*Merge*) Horizontal 130
7.5.3 Penggabungan Sel (*Merge*) Vertikal 131

7.6 Menyisipkan Gambar 131

7.7 Membuat Daftar Isi 133

7.8 Membuat Daftar Pustaka 133

7.8.1 Langsung pada *File* Utama 133

7.8.2 Dengan Menggunakan BibTeX 135

1 | *Density Functional Theory*

OLEH: EDDWI HESKY HASDEO

1.1 Pemodelan Material dengan Komputasi Prinsip Pertama

Komputasi prinsip pertama (*first-principles calculation*) disebut juga *ab initio* (baca: abinisio) merupakan metode teoretik untuk menghitung suatu fenomena fisika dengan parameter *fitting* seminimal mungkin. Dalam hal ini kita akan berfokus pada komputasi struktur elektronik material. Mekanika kuantum adalah teori fisika paling konsisten untuk menggambarkan fenomena mikroskopis yang berkaitan dengan interaksi elektron dengan potensial atom-atom dalam kristal. Akibat interaksi tersebut, kita mengenal ada material bersifat insulator, semi-konduktor, dan konduktor berdasarkan hasil perhitungan pita energi. Selain itu, kita juga dapat menentukan mode-mode getar dari atom dalam kristal yang terkuantisasi dengan nama fonon. Dengan mengetahui pita energi dan dispersi fonon dari suatu material, kita dapat menentukan sifat optis, sifat transpor, sifat mekanik, sifat termodinamika, dan sifat lainnya.

Dalam perhitungan sifat material, kita mengasumsikan bahwa elektron berada pada rezim kuantum tetapi kecepatannya tidak mendekati kecepatan cahaya (tidak relativistik) sehingga kita dapat menggunakan persamaan Schrödinger . Mungkin Anda pernah menyelesaikan persamaan Schrödinger untuk satu nukleus dan satu elektron pada atom hidrogen. Persamaan Schrödinger tiga dimensi (3D) tersebut diselesaikan dengan menggunakan koordinat bola dan separasi variabel dengan teknik matematika yang cukup kompleks. Anda dapat bayangkan bagaimana mungkin kita dapat menyelesaikan persamaan Schrödinger yang melibatkan 10^{23} elektron dan atom.

Felix Bloch menolong kita untuk menyederhanakan permasalahan ini untuk kristal. Di dalam kristal, terdapat periodisitas dengan simetri tertentu. Dengan memiliki struktur terkecil yang kita sebut *unit cell* dan jarak antar *unit cell* yang ditentukan oleh vektor translasi, kita dapat mereproduksi struktur kristal dengan *copy*, *paste*, dan menggeser *unit cell* dengan vektor translasi. Fisika yang melibatkan 10^{23} elektron dan atom tadi, dapat disederhanakan menjadi beberapa atom dan elektron saja di dalam *unit cell* .

Tentunya permasalahan sudah lebih sederhana, tapi masih belum terlalu sederhana. Persamaan Schrödinger yang melibatkan dua elektron saja misalnya, masih tetap kompleks. Terdapat model sederhana untuk mempelajari hal ini, yakni model Helium atau Hidrogen dengan dua elektron. Karena elektron merupakan partikel identik yang tunduk pada prinsip Pauli (tidak berkeadaan sama), pertukaran antar dua elektron menghasilkan fase negatif pada fungsi gelombang dua elektron tersebut. Ternyata sifat pertukaran antar elektron ini memberikan koreksi pada interaksi Coulomb yang disebut dengan *exchange interaction*. Koreksi interaksi Coulomb dari interaksi *exchange* ini ternyata sensitif terhadap susunan spin elektron yang menyebabkan terjadinya fenomena magnetisme. Pada material dengan banyak elektron, terdapat juga kemungkinan interaksi kuat (*correlation*) yang mengubah sifat elektronik dari material ini.

Kohn dan Hohenberg[1] membuat suatu hipotesis bahwa persamaan Schrödinger banyak elektron dapat disederhanakan menjadi persamaan Schrödinger satu elektron dengan menganggap bahwa interaksi *exchange* dan *correlation* adalah suatu fungsi dari kerapatan elektron. Itulah yang mendasari komputasi prinsip pertama atau *ab initio* disebut juga dengan *density functional theory* (DFT). Hamiltonian dari persamaan Schrödinger untuk satu elektron yang potensial interaksinya dituliskan sebagai suatu kerapatan disebut dengan Kohn-Sham Hamiltonian. Dengan melakukan tebakan awal suatu densitas elektron, kita dapat menghitung energi dan fungsi gelombang. Densitas adalah jumlah kuadrat dari fungsi gelombang. Dengan densitas yang baru kita melakukan iterasi perhitungan persamaan Schrödinger untuk meng-*update* fungsi gelombang. Akhirnya perhitungan akan selesai bila fungsi gelombang dan densitas sudah *self-consistent*.

Rentang waktu perkembangan komputasi prinsip pertama dapat diurutkan sebagai berikut:

1964	Teorema Hohenberg-Kohn dan formulasi Kohn-Sham
1972	Ekstensi relativistik untuk DFT
1980	Aproksimasi kerapatan lokal (LDA) untuk *exchange* dan *correlation*
1984	DFT bergantung waktu
1985	*Molecular dynamics* prinsip pertama
1986	Koreksi *quasiparticle* untuk insulator
1987	*Density functional perturbation theory*
1991	DFT dengan koreksi Hubbard
1996	*Generalized gradient approximation*

Dua paper penting yang mendasari prinsip DFT yaitu Hohenberg & Kohn, Phys. Rev. 136, B864 (1964) dan Kohn & Sham, Phys. Rev. 140, A1133 (1965) telah dikutip sebanyak 1500 dan 1000 kali setiap tahunnya. Paper-paper yang memberikan impak penting pada perkembangan DFT atau memakai prinsip DFT untuk menghasilkan teori atau *software* baru umumnya dikutip hingga puluhan ribu kali, seperti QUANTUM ESPREESO yang akan dibahas di BAB selanjutnya. DFT dipakai untuk melakukan prediksi maupun menjelaskan

[1] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

fenomena-fenomena fisika di berbagai material. Penggunanya cukup heterogen, mulai dari peneliti profesional hingga mahasiswa praktik.

DFT sangat populer karena memiliki keunggulan sebagai berikut:

- MUDAH DITRANSFER Kode atau metode DFT yang sama dapat dipakai untuk material yang berbeda,
- SEDERHANA Persamaan Kohn-Sham menyederhanakan Persamaan Schrödinger untuk elektron banyak,
- RELIABEL Dapat memprediksi properti material dengan akurasi tinggi hingga bahkan sebelum eksperimen,
- MUDAH DIBAGIKAN DFT telah menjadi perusahaan global (pengembangan *software* dengan kolaborasi global dan *open source*), dan
- PLATFORMNYA KOKOH Sering kali kelemahan pada DFT dapat diatasi dengan mengembangkan metode yang lebih kompleks, tetapi tetap berdasar pada DFT. Contoh: Metoda GW.

1.2 Persamaan Schrödinger Banyak Partikel

1.2.1 Persamaan Schrödinger untuk Atom Hidrogen

Pada BAB ini, kita akan membahas secara lebih konkret mengenai persamaan Schrödinger banyak partikel yang sempat diulas secara singkat pada BAB sebelumnya. Kita fokus pada sistem kristal yang terdiri atas 10^{23} elektron dan inti atom. Tetapi sebelum melangkah ke permasalahan yang terlalu kompleks, kita dapat membahas Persamaan Schrödinger untuk atom Hidrogen terlebih dahulu. Atom Hidrogen terdiri atas satu elektron dan satu proton, dituliskan sebagai berikut:

$$\underbrace{-\frac{\hbar^2}{2m_e}\nabla^2}_{\text{Energi kinetik}}\psi(\mathbf{r})\ \underbrace{-\frac{e^2}{4\pi\epsilon_0}\frac{1}{|\mathbf{r}|}}_{\text{Energi potensial}}\ \psi(\mathbf{r}) = E_{\text{tot}}\psi(\mathbf{r}). \tag{1.1}$$

$\hbar$ adalah tetapan Planck tereduksi ($h/2\pi$), m_e adalah massa elektron, e ($-e$) adalah muatan elektron (proton), ϵ_0 adalah permitivitas vakum, $\psi(\mathbf{r})$ adalah fungsi gelombang. Persamaan 1.1 mengasumsikan massa proton jauh lebih besar daripada m_e sehingga proton dapat dianggap diam terhadap elektron (tidak mempunyai energi kinetik). Persamaan ini melibatkan simetri bola, sehingga operator Laplacian ∇^2 harus dituliskan ke dalam koordinat bola dengan koordinat r merupakan jarak absolut inti terhadap elektron, θ sudut polar, dan ϕ sudut azimuthal. Setelah melakukan separasi variabel, kita menyelesaikan persamaan nilai eigen hingga didapatkan solusi:

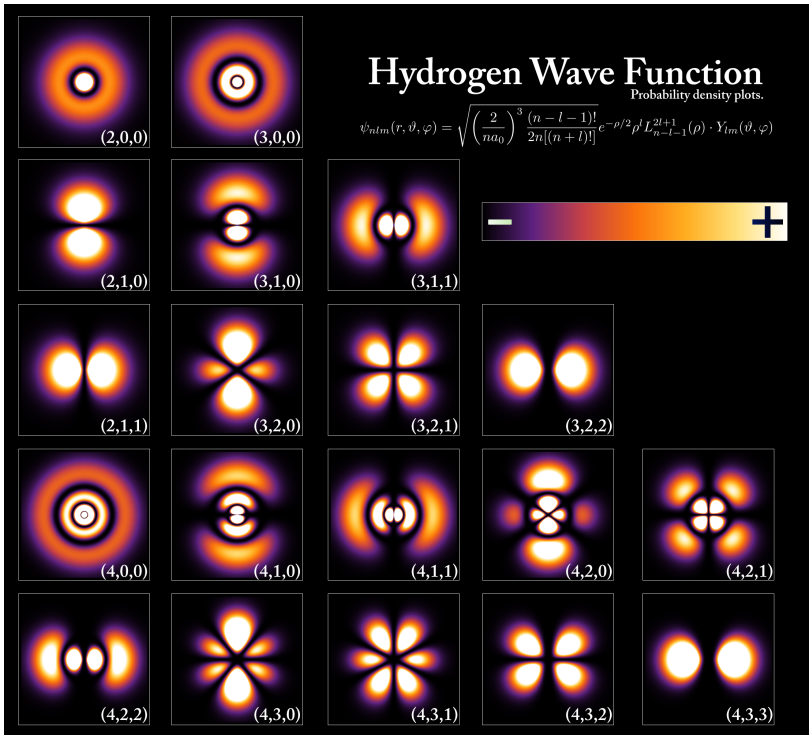
$$\psi_{nlm}(r,\theta,\phi) = \sqrt{\left(\frac{2}{na_0}\right)^3\frac{(n-l-1)!}{2n(n+l)!}}e^{-\rho/2}\rho^l L_{n-l-1}^{2l+1}(\rho)Y_l^m(\theta,\phi), \tag{1.2}$$

dengan

$$\rho = \frac{2r}{na_0}, \quad a_0 = \frac{4\pi\epsilon_0\hbar^2}{m_e e^2},$$

dimana, a_0 adalah radius Bohr, n, l, m adalah berturut-turut bilangan kuantum utama, azimuth, dan magnetik; $L_f^k(r)$ adalah polinomial Laguerre tergeneralisasi dengan derajad f dan orde ke k; dan $Y_l^m(\theta, \phi)$ adalah fungsi harmonik bola dengan derajad l dan orde ke m. Bilangan kuantum utama dapat bernilai $n = 1, 2, 3, \cdots$, bilangan kuantum azimuth dapat bernilai $l = 0, 1, 2, \cdots, n-1$ disebut juga $s, p, d, f, g, \cdots$, dan $m = -l, \cdots, l$. Solusi energinya didapatkan sebagai berikut:

$$E_n = -\frac{m_e e^4}{2(4\pi\epsilon_0)^2\hbar^2}\frac{1}{n^2} = \frac{-13.6}{n^2}\text{eV}. \tag{1.3}$$



Sumber: https://en.wikipedia.org/wiki/Atomic_orbital.

Penting untuk diingat bahwa spektrum energi atom hidrogen hanya bergantung pada bilangan kuantum utama n. Untuk n yang sama dan l dan m yang berbeda, E_n terdegenerasi. Tentu saja degenerasi ini hanya berlaku untuk atom hidrogen dan atom yang mirip hidrogen (misal golongan alkali). Untuk atom-atom lain yang lebih dari satu elektron, degenerasi ini akan terbuka.

1.2.2 Persamaan Schrödinger Banyak Partikel

Persamaan Schrödinger untuk satu elektron dan satu proton ternyata sudah melibatkan matematika pada level yang *advanced*. Kini kita coba menambah permasalahan sedikit lebih

rumit dengan mengasumsikan ada 3 elektron. Fungsi gelombang Schrödinger untuk tiga elektron kini menjadi seperti berikut:

$$\psi(\mathbf{r}) \to \Psi(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3), \tag{1.4}$$

Probabilitas untuk menemukan elektron nomor 1 pada posisi $\mathbf{r}$ adalah

$$\text{prob}(\mathbf{r}_1 = \mathbf{r}) = \int |\Psi(\mathbf{r}, \mathbf{r}_2, \mathbf{r}_3)|^2 d\mathbf{r}_2 d\mathbf{r}_3. \tag{1.5}$$

Karena elektron merupakan partikel tidak terbedakan, kerapatan elektron pada posisi $\mathbf{r}$ adalah:

$$\begin{aligned} n(\mathbf{r}) &= \text{prob}(\mathbf{r}_1 = \mathbf{r}) + \text{prob}(\mathbf{r}_2 = \mathbf{r}) + \text{prob}(\mathbf{r}_3 = \mathbf{r}) && (1.6) \\ &= 3 \int |\Psi(\mathbf{r}, \mathbf{r}_2, \mathbf{r}_3)|^2 d\mathbf{r}_2 d\mathbf{r}_3. && (1.7) \end{aligned}$$

Kini kita lihat bagaimana bentuk Hamiltonian untuk sistem banyak atom. Energi kinetik elektron dan inti atom adalah:

$$-\sum_{i=1}^{N} \frac{\hbar^2}{2m_e} \nabla_i^2 - \sum_{I=1}^{M} \frac{\hbar^2}{2M_I} \nabla_I^2. \tag{1.8}$$

Energi potensial, tolakan elektron-elektron:

$$\frac{1}{2} \sum_{i \neq j} \frac{e^2}{4\pi\epsilon_0} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}, \tag{1.9}$$

faktor $1/2$ dipakai untuk menghindari perhitungan ganda untuk interkasi i, j dengan j, i yang cukup dihitung sekali saja. Energi potensial, tolakan nukleus-nukleus:

$$\frac{1}{2} \sum_{I \neq J} \frac{e^2}{4\pi\epsilon_0} \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|}. \tag{1.10}$$

Energi potensial, tarikan elektron-nukleus:

$$-\sum_{i,I} \frac{e^2}{4\pi\epsilon_0} \frac{Z_I}{\mathbf{r}_i - \mathbf{R}_I}. \tag{1.11}$$

Kini bila kita ramu semua, kita akan mendapatkan persamaan Schrödinger banyak partikel sebagai berikut:

$$\begin{aligned} \Bigg[-\sum_{i=1}^{N} \frac{\hbar^2}{2m_e} \nabla_i^2 - \sum_{I=1}^{M} \frac{\hbar^2}{2M_I} \nabla_I^2 + \frac{1}{2} \sum_{i \neq j} \frac{e^2}{4\pi\epsilon_0} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \\ + \frac{1}{2} \sum_{I \neq J} \frac{e^2}{4\pi\epsilon_0} \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|} - \sum_{i,I} \frac{e^2}{4\pi\epsilon_0} \frac{Z_I}{\mathbf{r}_i - \mathbf{R}_I} \Bigg] \Psi &= E_{\text{tot}} \Psi \end{aligned} \tag{1.12}$$

Penulisan persamaan Schrödinger di atas akan lebih ramping bila kita perkenalkan sistem satuan Hartree.

- MASSA dalam satuan m_e, massa elektron,
- PANJANG dalam satuan a_0, radius Bohr, dan
- ENERGI dalam satuan $e^2/4\pi\epsilon_0 a_0$, energi Hartree.

Kini persamaan Schrödinger dapat dituliskan menjadi

$$\left[-\sum_{i=1}^{N}\nabla_i^2-\sum_{I=1}^{M}\frac{1}{2M_I}\nabla_I^2+\frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}+\frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|}-\sum_{i,I}\frac{Z_I}{\mathbf{r}_i-\mathbf{R}_I}\right]\Psi=E_{\text{tot}}\Psi. \tag{1.13}$$

Terlihat ramping karena dapat dituliskan ke dalam satu baris. Kini, kita melakukan aproksimasi bahwa massa nukeus $M_I=\infty$ sangat besar dibanding m_e. Selain itu, jarak antar nukleus $|\mathbf{R}_I-\mathbf{R}_J|$ dapat dianggap sangat besar, sehingga dapat disederhanakan sebagai berikut:

$$\left[-\sum_{i=1}^{N}\nabla_i^2+\frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}-\sum_{i,I}\frac{Z_I}{\mathbf{r}_i-\mathbf{R}_I}\right]\Psi=E_{\text{tot}}\Psi. \tag{1.14}$$

Ini sudah lebih sederhana, tetapi tetap belum sangat sederhana. Contohnya, bila kita ingin menyelesaikan persamaan Schrödinger banyak partikel untuk silikon yang mempunyai 8 elektron valensi. Kristal Si memiliki supercell berukuran $a=5$. Kita mendiskritkan jarak dengan $\Delta x\sim 0.1$. Di dalam tiga dimensi terdapat jumlah grid : $N_p=(a^3/4)/\Delta x^3\sim 40000$. Total fungsi gelombang akan memuat $\Psi=40000^8$ atau setara 10^{26} Terabyte. Perhitungan persamaan Schrödinger banyak partikel untuk susunan kristal paling sederhana seperti Si tidak akan sanggup dilakukan oleh komputer manapun.

Oleh sebab itu kita perlu melakukan suatu aproksimasi supaya kita dapat mengubah permasalahan banyak elektron menjadi permasalahan satu elektron. Pertama, kita coba untuk tidak menganggap interaksi antar elektron

$$\left[-\sum_{i=1}^{N}\nabla_i^2+\cancel{\frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}}-\sum_{i,I}\frac{Z_I}{\mathbf{r}_i-\mathbf{R}_I}\right]\Psi=\hat{H}_0(\mathbf{r})\Psi=E_{\text{tot}}\Psi. \tag{1.15}$$

Dengan tidak menganggap interaksi antar elektron ini, fungsi gelombang banyak partikel dapat dituliskan dengan lebih eksplisit:

$$\Psi(\mathbf{r}_1,\mathbf{r}_2,\cdots,\mathbf{r}_N)=\phi_1(\mathbf{r}_1)\cdots\phi_N(\mathbf{r}_N) \tag{1.16}$$

Persamaan Schrödinger untuk satu elektron menghasilkan

$$\hat{H}_0(\mathbf{r})\phi_i(\mathbf{r})=\varepsilon_i\phi_i(\mathbf{r}) \tag{1.17}$$

Sehingga, energi totalnya adalah

$$E=\varepsilon_1+\cdots+\varepsilon_N \tag{1.18}$$

Kita ambil contoh kasus untuk dua elektron ($Z = 2$) saling independen pada atom helium. Persamaan Schrödinger ditulis sebagai:

$$-\frac{1}{2}\nabla^2\phi(\mathbf{r}) - \frac{2}{|\mathbf{r}|}\phi(\mathbf{r}) = \varepsilon\phi(\mathbf{r}) \tag{1.19}$$

Solusi untuk energi terendah adalah:

$$\phi_{1s}(\mathbf{r}) = \frac{2^{3/2}}{\sqrt{\pi}}e^{-2|\mathbf{r}|}, \quad E_{1s} = -\frac{2^2}{2} \tag{1.20}$$

Pesamaan gelombang banyak partikel pada keadaan dasar adalah:

$$\Psi(\mathbf{r}_1, \mathbf{r}_2) = \phi_{1s}(\mathbf{r}_1)\phi_{1s}(\mathbf{r}_2) = \frac{8}{\pi}\exp[-2(|\mathbf{r}_1| + |\mathbf{r}_2|)] \tag{1.21}$$

Total energi pada keadaan dasar adalah:

$$E_{\text{tot}} = 2E_{1s} = -4 \text{ Ha} = -108.8 \text{ eV} \tag{1.22}$$

Kerapatan dua elektron pada aprosimasi dua elektron tak berinteraksi adalah:

$$n(\mathbf{r}) = 2\int |\Psi(\mathbf{r}, \mathbf{r}_2)|^2 d\mathbf{r}_2 = 2\int |\phi_1(\mathbf{r})|^2|\phi_2(\mathbf{r})|^2 d\mathbf{r}_2 = 2|\phi_1(\mathbf{r})|^2 \tag{1.23}$$

Sebenarnya persamaan gelombang 2 elektron harusla dituliskan sebagai kombinasi anti-simetrik terhadap pertukaran spin dan ruang $\mathbf{r}$ yang disebut determinan Slater. Hal ini dilakukan untuk memenuhi azas larangan Pauli. Kita menuslikan untuk 2 elektron tak berspin sebagai berikut

$$\Psi(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{\sqrt{2}}[\phi_1(\mathbf{r}_1)\phi_2(\mathbf{r}_2) - \phi_1(\mathbf{r}_2)\phi_2(\mathbf{r}_1)] \tag{1.24}$$

Densitas elektron untuk persamaan gelombang ini dapat ditulisakan sebagai:

$$n(\mathbf{r}) = 2\int |\Psi(\mathbf{r}, \mathbf{r})2)|^2 d\mathbf{r}_2 = |\phi_1(\mathbf{r})|^2 + |\phi_2(\mathbf{r})|^2 \tag{1.25}$$

Kerapatan elektron dapat digunakan untuk menghitung besarnya medan elektrostatik yang dihasilkan oleh elektron

$$\left[-\frac{1}{2}\nabla^2 + V_n(\mathbf{r})\right]\phi_i(\mathbf{r}) = \varepsilon_i\phi_i(\mathbf{r})$$

$$n(\mathbf{r}) = \sum_i |\phi_i(\mathbf{r})|^2 \tag{1.26}$$

$$V_H(\mathbf{r}) = \int d\mathbf{r}'\frac{n(\mathbf{r}'}{|\mathbf{r} - \mathbf{r}'|} \tag{1.27}$$

$$\left[-\frac{1}{2}\nabla^2 + V_n(\mathbf{r}) + V_H(\mathbf{r})\right]\phi_i^{\text{baru}}(\mathbf{r}) = \varepsilon_i^{\text{baru}}\phi_i(\mathbf{r})^{\text{baru}}, \tag{1.28}$$

$\phi(\mathbf{r})^{\text{baru}}$ yang didapatkan dari pers. (1.28) akan menjadi input untuk menghitung $n(\mathbf{r})$ yang baru pada pers. (1.26). Ini adalah persamaan *self-consistent* yang dikembangkan oleh Hartree pada tahun 1928.[2] Kekuatan dari persamaan ini adalah tidak memerlukan fungsi gelombang partikel banyak akan tetapi ia memerlukan solusi iteratif yang akan berhenti bila kriteria konvergensi terpenuhi.

Aproksimasi Hartree tidak mengikutsertakan batasan bahwa funsi gelombang elektron haruslah anti-simetrik terhadap pertukaran (*exchange*) posisi dan spin sesuai azas larangan Pauli: $\Psi(\mathbf{r}_1, \mathbf{r}_2) = -\Psi(\mathbf{r}_2, \mathbf{r}_1)$. Dengan memasukkan batasan *exchange* ini, kita mendapatkan tambahan kontribusi *Fock exchange*:

$$\left[-\frac{1}{2}\nabla^2 + V_n(\mathbf{r}) + V_H(\mathbf{r})\right]\phi_i^{\text{baru}}(\mathbf{r}) + \int d\mathbf{r}' V_x(\mathbf{r};\mathbf{r}')\phi_i(\mathbf{r}') = \varepsilon_i\phi_i(\mathbf{r}), \tag{1.29}$$

dengan

$$V_x(\mathbf{r};\mathbf{r}') = -\sum_{j\epsilon\text{occ}} \frac{\phi_j^*(\mathbf{r}')\phi_j(\mathbf{r})}{|\mathbf{r}-\mathbf{r}'|}.$$

Potensial Fock memastikan bahwa prinsip Pauli terpenuhi dengan konsekuensi dua elektron dengan spin yang sama saling tolak, namun dua spin yang berbeda saling tarik. Potensial Fock bersifat *non-local*; dapat lihat dari bentuk $V_x(\mathbf{r}, \mathbf{r}')$ yang terdiri atas dua koordinat.

Sejauh ini kita telah mengasumsikan bahwa elektron-elektron saling independen, dengan kata lain tidak terkorelasi. Sebagaimana cara menghitung probabilitas ditemukannya angka 2 dan 3 pada pelemparan dua buah dadu, probabilitas ditemukannya elektron pada posisi $\mathbf{r}_1$ dan elektron lain di posisi $\mathbf{r}_2$ adalah perlaian kedua probabilitas:

$$\text{prob}(\mathbf{r}_1, \mathbf{r}_2) = \text{prob}(\mathbf{r}_1) \times \text{prob}((\mathbf{r}_2) \tag{1.30}$$

Hal ini tentu tidak benar karena elektron saling tolak satu sama lain. KIta bisa bayangkan bila $\mathbf{r}_1$ dan $\mathbf{r}_2$ berdekatan tentunya nilainya tidak akan sama bila $\mathbf{r}_1$ dan $\mathbf{r}_2$ saling berjauhan. Dengan kata lain fungsi gelombang yang dituliskan sebagai determinan Slater tidak benar-benar merepresentasikan fungsi gelombang sesungguhnya. Akan tetapi karena perhitungan potensial *exchange* dan Hartree sudah menggunakan determinan SLater, kita harus tetap menggunakannya dengan menambahkan satu potensial lagi yang menentukan pengaruh korelasi,

$$\left[-\frac{1}{2}\nabla^2 + V_n + V_H + V_x + V_c\right]\phi_i = \varepsilon_i\phi_i. \tag{1.31}$$

[2] D. R. Hartree, Mathematical Proceedings of the Cambridge Philosophical Society **28**, 89 (1928).

1.3 Density Functional Theory

1.3.1 Fungsional dari Densitas

Density functional theory (DFT) adalah suatu teori tentang energi elektron yang merupakan fungsional dari densitas. Sebagai permulaan, kita akan membedakan dua istilah teknis matematika yaitu fungsi dan fungsional. Suatu fungsi, misalkan $g(x) = 3x^2$ memerlukan input x yang berupa angka dan akan menghasilkan nilai $g(x)$ yang berupa angka juga. Jadi, untuk nilai $x = 2$, kita mendapatkan $g(2) = 12$. Di samping itu, ada juga suatu fungsional yang membutuhkan input berupa fungsi untuk menghasilkan angka. Sebagai contoh $F[g] = \int_0^2 dx g(x)$, kita dapat menyebut F sebagai suatu fungsional karena membutuhkan input suatu fungsi $g(x)$. Dengan memasukkan $g(x) = 3x^2$, kita mendapatkan $F = 8$.

Di dalam persamaan Schrödinger kita tahu bahwa energi total adalah fungsional dari fungsi gelombang

$$\hat{H}\Psi = E\Psi, \quad E = \int d\mathbf{r}_1 \ldots d\mathbf{r}_N \Psi^* \hat{H} \Psi. \tag{1.32}$$

Untuk suatu fungsi gelombang umum, $\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N)$ kita dapat menuliskan fungsional $E[\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N)$. Pada tahun 1964, Hohenberg dan Kohn menunjukkan bahwa untuk Ψ di energi terendah, energi total juga merupakan fungsional dari kerapatan $E[n(\mathbf{r})]$.

Kita tuliskan kembali persamaan Schrödinger di BAB sebelumnya:

$$\left[\underbrace{-\sum_{i=1}^{N} \frac{1}{2}\nabla_i^2 + \frac{1}{2}\sum_{i \neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}}_{\hat{U}} + \sum_{i,I} \underbrace{\frac{-Z_I}{|\mathbf{r}_i - \mathbf{R}_I|}}_{V_n(\mathbf{r}_i)} \right] \Psi = E_{tot}\Psi. \tag{1.33}$$

Kita menuliskan fungsional energi menjadi lebih kompak:

$$E = \int d\mathbf{r}\ n(\mathbf{r}) V_n(\mathbf{r}) + \langle \Psi | \hat{U} | \Psi \rangle, \quad n(\mathbf{r}) = \sum_i |\phi_i(\mathbf{r})|^2, \quad \Psi = \Psi[\phi_1, \ldots, \phi_N]_{\text{determinan Slater}}. \tag{1.34}$$

Kita akan menunjukkan bahwa pada keadaan dasar, $n(\mathbf{r})$ menentukan secara unik besarnya $V_n(\mathbf{r})$. Pertama, kita anggap ada dua potensial V_1 dan V_2 untuk densitas yang sama. Dengan menyelesaikan persamaan Schrödinger banyak partikel, kita menemukan energi terendah dan fungsi gelombangnya, masing-masing E_1, Ψ_1 dan E_2, Ψ_2 secara berurutan. Karena Ψ_1 adalah keadaan dasar dari V_1, kita mendapat

$$\int d\mathbf{r} n(\mathbf{r}) V_1 + \langle \Psi_1 | \hat{U} | \Psi_1 \rangle = E_1, \tag{1.35}$$

Namun Ψ_1 bukanlah keadaan dasar dari V_2, maka kita mendapat energi yang lebih tinggi dari E_2,

$$\int d\mathbf{r} n(\mathbf{r}) V_2 + \langle \Psi_1 | \hat{U} | \Psi_1 \rangle > E_2, \tag{1.36}$$

Perbedaan energi diberikan oleh:

$$\int d\mathbf{r} n(\mathbf{r})(V_1 - V_2) > E_1 - E_2 \tag{1.37}$$

Dengan mengulangi langkah yang sama untuk Ψ_2, kita mendapat

$$\int d\mathbf{r} n(\mathbf{r})(V_2 - V_1) > E_2 - E_1 \tag{1.38}$$

Jumlahkan kedua persamaan tersebut menghasilkan kontradiksi, $0 > 0$. Hal ini berarti tidak mungkin terdapat V_1 dan V_2 yang berbeda untuk n yang sama. Dengan kata lain n menentukan nilai V_n secara unik. Hal ini merupakan kabar baik yang dapat menyederhanakan hitungan. Namun, kabar buruknya, bentuk fungsional dari energi masih tidak diketahui secara pasti.

Kita tuliskan kembali fungsional energi di atas:

$$E[n] = \int d\mathbf{r}\ n(\mathbf{r}) V_n(\mathbf{r}) + \frac{1}{2} \int d\mathbf{r} d\mathbf{r}' \frac{n(\mathbf{r}) n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} - \sum_i \int d\mathbf{r} \phi_i^*(\mathbf{r}) \frac{\nabla^2}{2} \phi_i(\mathbf{r}) + E_{xc}[n] \tag{1.39}$$

Kohn dan Sham (1965) mengusulkan untuk menuliskan densitas elektron seolah elektron bebas (tidak berkorelasi)

$$n(\mathbf{r}) = \sum_{i \in \text{occ}} |\phi_i(\mathbf{r})|^2. \tag{1.40}$$

Kita akan menemukan energi terendah dengan menggunakan syarat variasi:

$$\frac{\delta E}{\delta n} = 0, \quad \text{dan} \quad \langle \phi_i | \phi_j \rangle = \delta_{ij}. \tag{1.41}$$

Hal ini akan menghasilkan persamaan Kohn-Sham:

$$\left[-\frac{1}{2} \nabla^2 + v_n(\mathbf{r}) + V_H(\mathbf{r}) + V_{xc}(\mathbf{r}) \right] \phi_i(\mathbf{r}) = \epsilon_i \phi_i(\mathbf{r}), \quad V_{xc} = \frac{\delta E_{xc}}{\delta n} \tag{1.42}$$

Untuk menentukan potensial *exchange* dan korelasi, kita menggunakan suatu pendekatan. Pendekatan paling sederhana adalah *local density approximation* (LDA). Kita dapat menghitung E_{xc} menggunakan teori Thomas-Fermi. Energi exchange Hartree-Fock dapat dituliskan sebagai:

$$E_{xc}[n] = -\frac{1}{4} \int d\mathbf{r}_1 d\mathbf{r}_2 \frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|} |n(\mathbf{r}_1, \mathbf{r}_2)|^2 \tag{1.43}$$

Fungsi gelombang elektron bebas adalah

$$\phi(\mathbf{r}) = \frac{1}{\sqrt{V}} e^{i\mathbf{k}\cdot\mathbf{r}} \tag{1.44}$$

Kerapatan elektron adalah:

$$\begin{aligned} n(\mathbf{r}_1,\mathbf{r}_2) = \frac{2}{V}\sum_{\mathbf{k}\in\text{occ}} e^{i\mathbf{k}\cdot(\mathbf{r}_1-\mathbf{r}_2)} &= \frac{1}{4\pi^3}\int e^{i\mathbf{k}\cdot(\mathbf{r}_1-\mathbf{r}_2)}d^3\mathbf{k}, \\ &= \frac{1}{4\pi^3}\int_0^{k_F} k^2dk\int\int e^{ikr_{12}\cos\theta}\sin\theta d\theta d\varphi \\ &= 3n(\mathbf{r})\left[\frac{\sin t - t\cos t}{t^3}\right], \end{aligned} \tag{1.45}$$

dengan

$$t = k_F(|\mathbf{r}_1-\mathbf{r}_2|), \quad n(\mathbf{r}) = \frac{k_F^3}{3\pi^2}, \quad \mathbf{r} = \frac{1}{2}(\mathbf{r}_1+\mathbf{r}_2).$$

Substtusi persamaan Pers.(1.45) ke Pers. (1.43) kita mendapat:

$$E_{xc}[n] = -9\pi\int\frac{n^2(\mathbf{r})}{k_F^2}d\mathbf{r}\left[\int_0^\infty\frac{(\sin t - t\cos t)^2}{t^5}dt\right] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3}\int n^{4/3}(\mathbf{r})d\mathbf{r}. \tag{1.46}$$

Maka, potensial LDA dapat dituliskan sebagai:

$$V_{xc}^{\text{LDA}} = \frac{\delta E_{xc}^{\text{LDA}}}{\delta n} = -\left(\frac{3}{\pi}\right)^{1/3} n^{1/3}(\mathbf{r}). \tag{1.47}$$

1.3.2 Perhitungan *self-consistent field* (SCF)

Kita ketahui bahwa interaksi antar elektron menghasilkan medan listrik $-\nabla V_H$ yang menentukan kerapatan elektron $n(\mathbf{r})$. Kerapatan elektron berkaitan dengan fungsi gelombang yang kita butuhkan untuk memecahkan persamaan Schrödinger untuk mendapatkan nilai energinya. Langkah-langkah penentuan nilai energi dengan iterasi sehingga didapatkan energi yang konsisten disebut perhitungan *self-consistent field* (SCF). Berikut adalah langkah-langkah SCF

1. Mendefinisikan struktur atom sehingga didapat potensial elektron-ion $V_n(\mathbf{r}) = -\sum_I \frac{Z_I}{|\mathbf{r}-\mathbf{R}_I|}$.
2. Membuat tebakan awal densitas elektron $n(\mathbf{r})$.
3. Menghitung potensial Hartree $\nabla^2 V_H(\mathbf{r}) = -4\pi n(\mathbf{r})$, dan potensial exchange korelasi $V_{xc} = \frac{\delta E_{xc}[n](\mathbf{r})}{\delta n}$.
4. Mendapatkan potensial total : $V_{tot}(\mathbf{r}) = V_n(\mathbf{r}) + V_H(\mathbf{r}) + V_{xc}(\mathbf{r})$.
5. Menyelesaikan persamaan Schrödinger : $\left[-\frac{1}{2}\nabla^2 + V_{tot}(\mathbf{r})\right]\phi_i(\mathbf{r}) = \epsilon_i\phi_i(\mathbf{r})$.
6. Menentukan kerapatan yang baru : $n(\mathbf{r}) = \sum_i |\phi_i(\mathbf{r})|^2$.

7. Cek apakah (energi baru = energi lama) ? Jika iya, maka perhitungan selesai. Jika tidak, maka kembali ke langkah ke 3 dengan menggunakan densitas yang baru.

2 | Quantum Espresso

OLEH: SASFAN ARMAN WELLA

QUANTUM ESPRESSO (QE)[1] merupakan *software* yang biasa digunakan untuk simulasi kuantum berbagai macam material berdasarkan *density functional theory* (DFT). QE menggunakan *plane-wave pseudopotential* dan *projector-augmented-wave approaches* yang memungkinkan penggunanya untuk menggunakan sistem yang kecil sebagai representasi dari sistem yang besar, khususnya untuk sistem yang berupa kristal. Berbeda dengan *software* sejenis, seperti VASP, Siesta, CASTEP, DMol3, dan lain lain, yang membutuhkan lisensi penggunaan, QE merupakan *software open-source* yang tidak berbayar atau gratis, sehingga banyak digunakan dalam berbagai penelitian.

2.1 Instalasi Quantum Espresso

Pertama, kita perlu melakukan instalasi QE di komputer atau *work station* yang akan kita gunakan. *Software* ini pada dasarnya hanya dapat dijalankan di komputer berbasis Linux. Namun, pengguna komputer berbasis Windows tidak perlu khawatir, sebab kini Windows (untuk versi Windows 10 ke atas) telah menyediakan subsistem untuk menjalankan Linux, atau yang dikenal dengan WSL (Windows Subsystem for Linux)[2]. Atau, pengguna komputer berbasis Windows dan juga Macintosh dapat menginstal VirtualBox[3], kemudian menginstal Linux di dalamnya.

Dengan asumsi bahwa Linux telah terinstal dengan baik, mari kita mulai menginstal QE. Langkah pertama, kita install perangkat dasarnya terlebih dulu.

```
$ sudo apt-get install build-essential fftw3-dev gfortran
```

[1] P. Giannozzi *et al.*, J.Phys.: Condens.Matter **21**, 395502 (2009)
[2] https://docs.microsoft.com/en-us/windows/wsl/install-win10
[3] https://www.virtualbox.org/

Berikutnya, kita lakukan instalasi LAPACK (Linear Algebra PACKage), yaitu paket standar yang berisi kumpulan operasi aljabar linier tingkat tinggi. Paket tersebut dapat kita unduh melalui tautan berikut: http://www.netlib.org/lapack/. LAPACK dapat diinstal dengan langkah-langkah sebagai berikut:

```
$ cd lapack-3.7.1
$ cp make.inc.example make.inc
$ make blaslib
$ make lapacklib
$ make tmglib
$ sudo cp librefblas.a /usr/local/lib/libblas.a
$ sudo cp liblapack.a /usr/local/lib/liblapack.a
$ sudo cp libtmglib.a /usr/local/lib/libtmg.a
```

Software pendukung lainnya yang sebaiknya kita instal ialah OpenMPI. *Software* ini berguna untuk memaksimalkan *CPU cores* yang tersedia pada komputer. Dengan begitu, kita bisa melakukan komputasi paralel ketika menjalankan QE. OpenMPI dapat kita unduh melalui tautan berikut: https://www.open-mpi.org/. *Software* ini diinstal dengan langkah-langkah sebagai berikut:

```
$ tar xvf openmpi-X.X.X.tar.gz
$ cd openmpi-X.X.X/
$ ./configure --prefix=/usr/local/openmpi-X.X.X
$ make install
```

Opsi `prefix` menunjukkan posisi dimana OpenMPI diinstal. Selanjut, kita perlu menambahkan 1 baris perintah tambahan pada `./bashrc`, sehingga komputer kita mengetahui dari mana OpenMPI dapat diakses.

```
$ export PATH=$PATH:/usr/local/openmpi-X.X.X/bin
```

Untuk memastikan apakah OpenMPI telah siap digunakan, maka kita dapat mengetikkan perintah berikut:

```
$ which mpirun
```

Apabila `path` letak OpenMPI telah muncul, artinya OpenMPI siap digunakan.

Setelah *software* dan paket pendukung telah terinstall dengan baik, selanjutnya kita melakukan instalasi untuk QE. *Installer* QE dapat diunduh melalui tautan berikut:

http://www.quantum-espresso.org/.

Setelah berhasil, lakukan langkah-langkah berikut:

```
$ tar -xzvf qe-X.X.tar.gz
$ cd qe-X.X
```

Cara lain untuk mendapatkan installer QE terbaru bisa dilakukan dengan perintah berikut:

```
$ git clone https://github.com/QEF/q-e.git
$ cd qe
```

Setelah installer berhasil diunduh, kita lanjutkan ke tahap konfigurasi.

```
$ ./configure
```

Pada tahap ini, pastikan bahwa hasil konfigurasinya berhasil mendeteksi bahwa sistem komputer kita mendukung untuk menjalankan QE secara paralel. Caranya, pastikan pesan di bawah ini muncul:

Parallel environment detected successfully.
Configured for compilation of parallel executables.

Tahap selanjutnya merupakan tahap yang membutuhkan waktu paling lama. Jadi, harap bersabar.

```
$ make all
```

Setelah selesai, kita kembali perlu menambahkan `path` QE ke dalam `./bashrc` dan QE siap untuk digunakan.

```
$ export PATH=$PATH:/home/username/qe-X.X/bin/
```

2.2 Input Data Quantum Espresso

Sebelum melakukan perhitungan, kita perlu membuat *file* input agar QE dapat memahami apa yang ingin kita hitung. Seperti yang bisa kita lihat pada Gambar 2.1, input data pada QE memiliki struktur khusus yang perlu dipatuhi. Secara umum, input QE terdiri dari beberapa blok, seperti `&CONTROL`, `&SYSTEM`, `&ELECTRONS`, dan sebagainya. Blok dengan tanda "[" dan "]" menunjukkan bahwa blok tersebut bersifat opsional, yang hanya dituliskan untuk perhitungan tertentu saja.

Terdapat sejumlah parameter yang bisa kita tentukan pada masing-masing blok. Apabila kita tidak menulistkan parameter tersebut di dalam *file* input, artinya kita menggunakan nilai *default* untuk parameter tersebut. Daftar parameter beserta penjelasannya dapat dilihat pada tautan berikut: https://www.quantum-espresso.org/Doc/INPUT_PW.html.

```
&CONTROL
  ...
/

&SYSTEM
  ...
/

&ELECTRONS
  ...
/

[ &IONS
  ...
 / ]

[ &CELL
  ...
 / ]

ATOMIC_SPECIES
 X  Mass_X  PseudoPot_X
 Y  Mass_Y  PseudoPot_Y
 Z  Mass_Z  PseudoPot_Z

ATOMIC_POSITIONS { alat | bohr | crystal | angstrom | crystal_sg }
  X 0.0  0.0  0.0  {if_pos(1) if_pos(2) if_pos(3)}
  Y 0.5  0.0  0.0
  Z 0.0  0.2  0.2

K_POINTS { tpiba | automatic | crystal | gamma | tpiba_b | crystal_b | tpiba_c | crystal_c }
if (gamma)
   nothing to read
if (automatic)
   nk1, nk2, nk3, k1, k2, k3
if (not automatic)
   nks
   xk_x, xk_y, xk_z,  wk
if (tpipa_b or crystal_b in a 'bands' calculation) see Doc/brillouin_zones.pdf

[ CELL_PARAMETERS { alat | bohr | angstrom }
   v1(1) v1(2) v1(3)
   v2(1) v2(2) v2(3)
   v3(1) v3(2) v3(3) ]
```

Gambar 2.1: Struktur input data QUANTUM ESPRESSO.

Pada tutorial ini, kita akan bahas beberapa parameter yang sering digunakan di dalam perhitungan.[4]

`&CONTROL`

Parameter	Keterangan	Nilai
`calculation`	Jenis kalkulasi yang hendak dilakukan.	'scf', 'nscf', 'bands', dll.
`restart_mode`	Kalkulasi awal atau lanjutan.	'from_scratch', 'restart'.
`nstep`	Jumlah step maksimum untuk iterasi ionik dan iterasi electronik.	**default**: 1 untuk scf, nscf, dan band; 50 untuk jenis perhitungan yang lain.
`outdir`	Direktori dimana *file* output akan dituliskan.	-
`pseudo_dir`	Direktori dimana *pseudopotential* disimpan.	-
`prefix`	Label perhitungan.	-
`tprnfor`	Untuk menampilkan nilai gaya pada *file* output.	`.true.` atau `.false.`
`tsress`	Untuk melibatkan perhitungan *stress* atau tidak.	`.true.` atau `.false.`
`forc_conv_thr`	Nilai *threshold* konvergensi ionik yang dinyatakan dalam satuan gaya.	**default**: 0.001

`&SYSTEM`

Parameter	Keterangan	Nilai
`ibrav`	Indeks kisi Bravais yang akan digunakan.	0 untuk *free structure*, 1 untuk *SC*, 2 untuk FCC, 3 untuk BCC, dll.
`celldm(i)`	Konstanta kisi dalam satuan Bohr.	i=1 (a) untuk kubus, i=1 (a) dan i=2 (c) untuk heksagonal, dll.
`nat`	Jumlah atom/ion.	-
`type`	Jumlah tipe atom/ion.	-
`ecutwfc`	Energi *cut-off* untuk fungsi gelombang (dalam Ry).	-
`ecutrho`	Energi *cut-off* untuk *charge density* dan potensial (dalam Ry).	**default**: $4\times$`ecutwfc`, untuk *ultrasoft* PP gunakan 8 sampai $12\times$`ecutwfc`.

[4] M. H. Mahyuddin dan L. Y. Ginting. MRS.Id-Bulletin **1**, 2013.

`occupations`	Okupansi pada *Fermi level*.	Gunakan 'smearing' untuk sistem metalik, dan 'fixed' untuk insulator.
`smearing`	Metoda untuk mengatasi permasalahan okupansi pada *Fermi level*.	'gaussian'=gaussian, 'mp'=Methfessel-Paxton, 'mv'=Marzari-Vanderbilt, dll.
`degauss`	Lebar *spreading* pada `smearing`	**default**: 0
`nspin`	Pilihan untuk *polarized calculation* (PC) atau bukan.	1=Non-PC, 2=PC *collinear*, 4=PC *non-collinear*.
`start_magnetization(i)`	Nilai awal magnetisasi untuk perhitungan yang melibatkan polarisasi spin.	-1 untuk *spin-down*, 1 untuk *spin-up*

`&ELECTRONS`

Parameter	Keterangan	Nilai
`electron_maxstep`	Jumlah iterasi maksimum pada iterasi elektronik (SCF).	**default**: 100
`conv_thr`	Nilai *threshold* konvergensi SCF.	**default**: 1.D-6
`diagonalization`	Algoritma untuk diagonalisasi matriks.	'david'=Davidson, 'cg'=Metoda *conjugated-gradient*.
`mixing_beta`	Bobot pencampuran *wavefunction* lama dan baru.	**default**: 0.7

`&IONS`

Parameter	Keterangan	Nilai
`ion_dynamics`	Metoda untuk relaksasi ionik.	'bfgs'=Algoritma quasi-Newton, 'damp'=Metoda quick-min Verlet.

`ATOMIC_SPECIES`

Parameter	Keterangan	Nilai
`X`	Label dari elemen atom yang digunakan.	C untuk Karbon, Si untuk Silikon, dll.
`Mass_X`	Massa atom yang digunakan (dalam amu).	Mis: C=12
`PseudoPot_X`	Pseudopotential yang sesuai dengan elemen atom yang digunakan.	Mis: `C.pz-bhs.UPF`

ATOMIC_POSITIONS

Parameter	Keterangan
alat	Kordinat Kartesian dalam unit parameter kisi a atau celldm(1).
bohr/angstrom	Kordinat Kartesian dalam unit Bohr atau Å.
crystal	Kordinat relatif terhadap CELL_PARAMETERS.

CELL_PARAMETERS: Hanya digunakan saat ibrav= 0. Jika ibrav $\neq 0$, maka QE akan menggunakan CELL_PARAMETERS bawaan yang sesuai dengan kisi Bravais yang digunakan.

K_POINTS

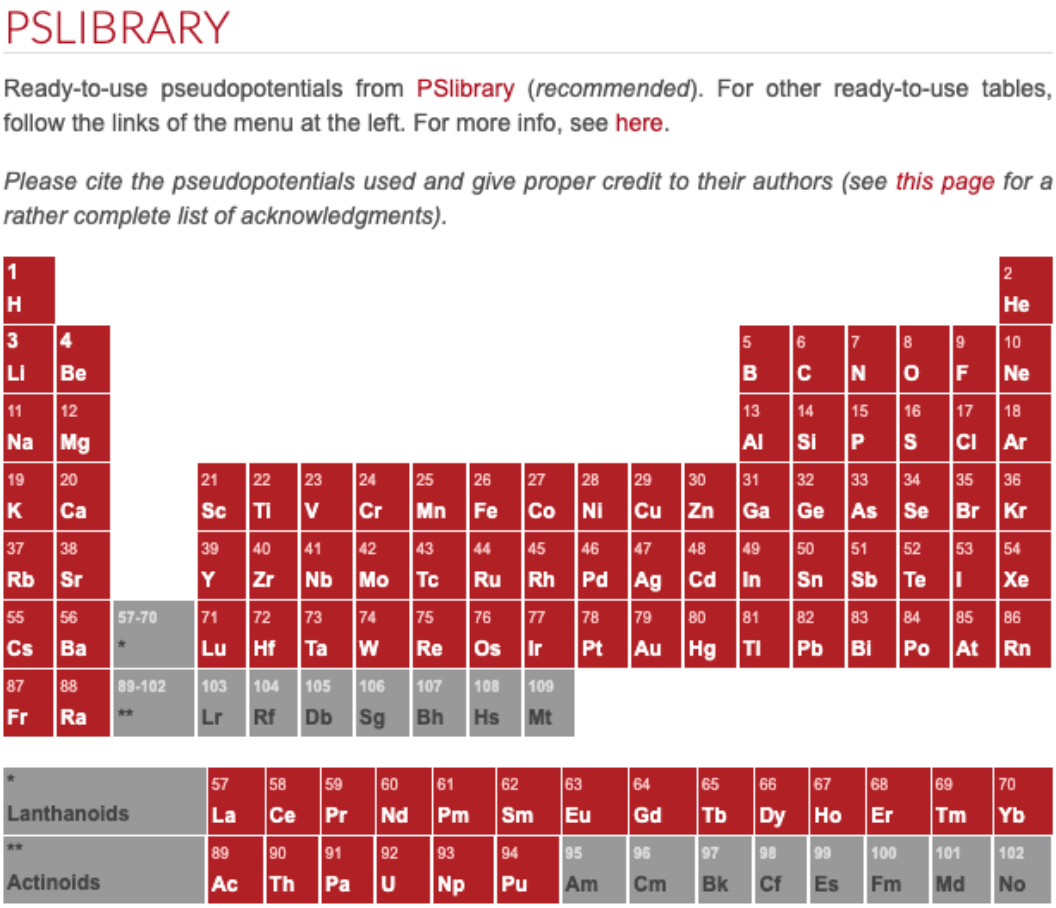
Parameter	Keterangan
automatic	Menciptakan grid Monkhorst-Pack secara otomatis sejumlah nk dengan *offset* sk.
tpiba	Membaca k-points pada koordinat Kartesian (dalam $2\pi/a$) untuk setiap titik yang ingin ditinjau dalam ruang respirok.
crystal	Sama seperti mode tpiba, tapi mode ini membaca k-points pada koordinat kristal.
tpiba_b	Pada dasarnya sama dengan tpiba, tetapi kita bisa melakukan penambahan $N-1$ titik secara otomatis di antara 2 titik yang ingin ditinjau.
crystal_b	Sama seperti mode tpiba_b, tapi mode ini membaca k-points pada koordinat kristal.

Berikut ini contoh penggunaan mode crystal dan crystal_b:

```
K_POINTS {crystal}
5
0.0 0.0 0.0    1.0
0.1 0.0 0.0    1.0
0.2 0.0 0.0    1.0
0.3 0.0 0.0    1.0
0.4 0.0 0.0    1.0
0.5 0.0 0.0    1.0
```

```
K_POINTS {crystal_b}
2
0.0 0.0 0.0    5.0
0.5 0.0 0.0    1.0
```

Bisa kita lihat bahwa dengan mode crystal_b kita hanya perlu menuliskan 2 titik (awal dan akhir) yang ingin dikaji. Setelah itu, sistem akan mencarikan $N-1$ titik (dalam hal ini $5-1=4$ titik) diantara titik $(0,0,0)$ dan $(\frac{1}{2},0,0)$. Jika kita menggunakan mode crystal, kita perlu menulis titik-titik tersebut secara manual.



Gambar 2.2: Tampilan *library pseudopotential* `Quantum Espresso`.

2.3 *Pseudopotential*

Selain *file* input, kita pun membutuhkan *file pseudopotential* (PP) sesuai dengan elemen atom yang kita gunakan. *File* tersebut dapat kita peroleh melalui tautan berikut:

https://www.quantum-espresso.org/pseudopotentials

Seperti yang ditunjukkan oleh Gambar 2.2, kita dapat mengunduh setiap PP yang dibutuhkan dengan mengklik menu yang tersedia. Atau, kita dapat *copy* tautan PP yang kita akan unduh, kemudian tuliskan perintah berikut pada terminal:

```
$ wget https://www.quantum-espresso.org/upf_files/(Nama PP).UPF
```

2.4 Parameter Konvergensi

Sebelum kita melakukan perhitungan dengan menggunakan QE, kita perlu mengecek konvergensi perhitungan kita dengan menentukan beberapa parameter konvergensi, seperti `ecutwfc`, `ecutrho`, dan `k-points`. Perlu disampaikan bahwa **sebagian besar** pembahasan tutorial ini berasal dari *Hands-on "An Introduction to DFT for Experimentalist"*[5] yang dibuat oleh Prof. Feliciano Guistino dan tim. Jadi, untuk bagi yang tertarik dengan tutorial yang yang dibuat oleh mereka, silahkan mengunjungi situs https://www.paradim.org.

2.4.1 Menentukan nilai `ecutwfc`

Kita mulai dengan parameter `ecutwfc`, atau yang dikenal dengan energi *cutoff* untuk *wavefunction*, merupakan salah satu parameter penting dalam konvergensi suatu perhitungan DFT yang berbasis *pseudopotentials* dan *planewaves*. Sebelumnya, mari kita siapkan *file* input (`silicon-1.scf.in`) untuk sistem silikon, seperti berikut:

```
&CONTROL
  calculation = "scf",
  prefix = "silicon",
  pseudo_dir = "./",
  outdir = "./"
/
&SYSTEM
  ibrav = 2,
  celldm(1) = 10.28,
  nat = 2,
  ntyp = 1,
  ecutwfc = 5.0,
/
&ELECTRONS
/
ATOMIC_SPECIES
  Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS {alat}
  Si 0.00 0.00 0.00
  Si 0.25 0.25 0.25
K_POINTS {automatic}
  4 4 4 1 1 1
```

[5] https://www.paradim.org/DFT_SS_2021

Di sini, kita menggunakan `ecutwfc` sebesar 5 Ry (1 Ry = 13.6058 eV). *File* input tersebut kemudian dijalankan dengan perintah berikut:

Untuk perhitungan secara tunggal (tidak secara paralel)

```
pw.x < silicon-1.scf.in > silicon-1.scf.out
```

Untuk perhitungan secara paralel

```
mpirun -np 4 pw.x < silicon-1.scf.in > silicon-1.scf.out
```

Perintah di atas mengandung arti bahwa *file* input `silicon-1.scf.in` akan dihitung dengan program `pw.h` dan hasilnya akan ditulis pada *file* output `silicon-1.scf.in`. Angka 4 setelah `-np` menunjukkan berapa jumlah *CPU cores* yang akan digunakan. Sebaiknya disesuaikan dengan jumlah *CPU cores* yang tersedia pada komputer masing-masing.

Setelah perhitungan selesai, maka pada bagian akhir `silicon-1.scf.out` akan muncul tulisan "JOB DONE". Tanpa harus membuka file output tersebut, kita ketikkan perintah berikut:

```
grep ! silicon-1.scf.out
```

Hasilnya, kita akan memperoleh tampilan berikut:

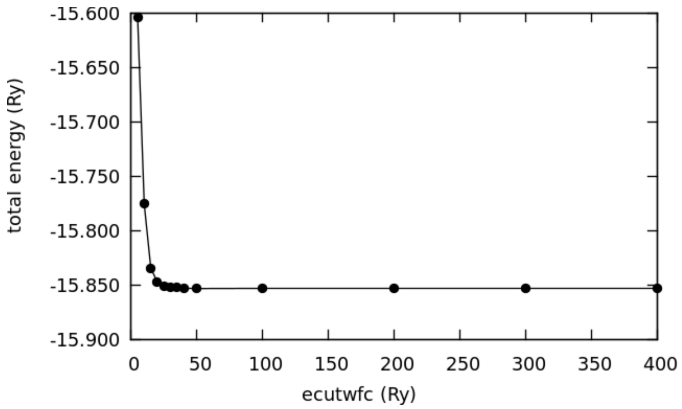
```
!    total energy              =      -15.60437807 Ry
```

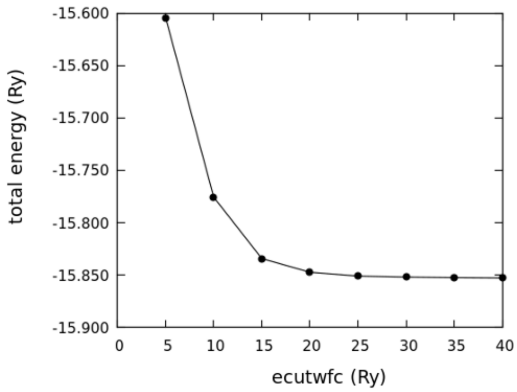
Di sini, kita memperoleh data berupa energi total hasil perhitungan DFT berdasarkan parameter-parameter yang kita gunakan. Perlu diperhatikan bahwa nilai mutlak dari energi total ini tidak sepenuhnya kita gunakan, sebab energi total ini tidak merepresentasikan kondisi vakum (sebagaimana yang kita lakukan ketika menyelesaikan persamaan Schrödinger untuk atom hidrogen). QE sendiri merupakan perhitungan quantum yang diperuntukkan untuk sistem yang periodik secara tak-hingga, seperti struktur kristal. Oleh karena itu, alih-alih menggunakan nilai mutlak dari energi total ini, yang biasa dilakukan ialah menggunakan perbedaan energi total agar *offset* yang terkandung di dalam hasil perhitungan dapat saling menghilangkan.

Selanjutnya, kita ulangi perhitungan kita dengan menggunakan `ecutwfc` yang berbeda: 5, 10, 15, 20, 25, 30, 35, 40, 50, 100, 200, 300, dan 400 Ry. Apabila sudah selesai, kita kumpulkan data energi total untuk setiap nilai `ecutwfc` yang kita gunakan, seperti berikut:

```
# ecutwfc (Ry) Energi Total (Ry)
  5 -15.60437792
  10  -15.77558475
  ... ...
  400 -15.85268613
```

Dan apabila kita plot dengan GNUPLOT[6] kita akan memperoleh grafik sebagai berikut:





Dari grafik tersebut, kita dapat melihat bahwa dengan menggunakan `ecutwfc` sebesar 25 Ry, energi totalnya hanya 15 meV/atom lebih tinggi dari `ecutwfc` sebesar 400 Ry. Nilai *error* ini cukup bisa ditolerir sehingga `ecutwfc` sebesar 25 Ry sudah cukup baik untuk kita gunakan dalam perhitungan kita.

2.4.2 Menentukan nilai `ecutrho`

Selanjutnya, kita tentukan nilai untuk `ecutrho`, atau nilai energi *cutoff* untuk *charge density*. Sebagaimana yang disarankan oleh QE, kita nilai `ecutrho` sebesar $4\times$ nilai `ecutwfc`. Namun, akan lebih baik jika kita melakukan pengecekan yang sama seperti ketika menentukan `ecutwfc`. Catatan lain, apabila kita menggunakan *Ultrasoft* PP, kita sebaiknya menggunakan nilai `ecutrho` sebesar $8\times$ sampai $12\times$ nilai `ecutwfc`.

2.4.3 Menentukan nilai `k-points`

Parameter lain yang tidak kalah penting, yaitu `k-points`. Seperti yang tertulis pada bagian akhir *file* input `silicon-1.scf.in` ($4\ 4\ 4\ 1\ 1\ 1$), kita membagi zona Brillouin (*sampling*) dari sistem tersebut dalam $4 \times 4 \times 4$ grid. Tiga angka selanjutnya ($1\ 1\ 1$) menunjukkan bahwa kita mengizinkan QE untuk melakukan pergeseran (*shifting*) sejauh setengah jarak antar grid di segala arah (x,y, dan z). Penggunaan *shifting* ini digunakan karena biasanya menghasilkan *sampling* yang lebih baik. Sebaliknya, kita bisa menuliskan ($0\ 0\ 0$) untuk mode *unshifted*.

Apabila kita melihat *file* output dari `silicon-1.scf.in` dan mencari kata kunci `'number of k points'`, kita akan memperoleh bahwa grid yang digunakan hanya 10 grid (bukan $4 \times 4 \times 4 = 64$ grid). Perbedaan ini dikarenakan QE bisa mengeliminasi grid yang simetri. Dengan kata lain, 10 grid yang digunakan dalam perhitungan hanya grid yang tidak ekuivalen.

[6]Pembahasan mengenai penggunaan GNUPLOT dapat dilihat di BAB 5 Modul ini.

Kini kita akan mengulangi perhitungan kita, dengan mengubah-ubah parameter `k-points`: (1 1 1 1 1 1), (2 2 2 1 1 1), (4 4 4 1 1 1), (8 8 8 1 1 1), 16 16 16 1 1 1) dan (16 16 16 1 1 1). Apabila sudah selesai, kita kumpulkan data energi total dan jumlah grid yang tidak ekuivalen untuk setiap nilai `ecutwfc` yang kita gunakan, seperti berikut:

Data untuk mode *shifted*

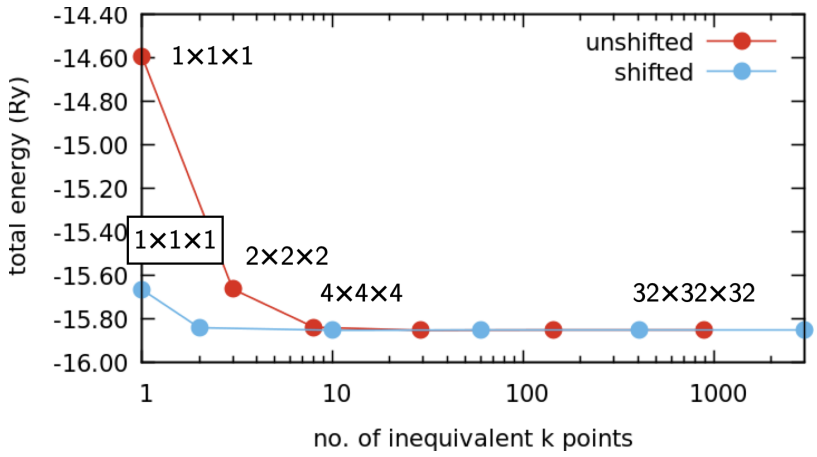
```
# k-points  Shift Energi total (Ry) Grid non-ekuivalen
  1 1 -15.66368659     1
  ... ... ...      ...
  32  1 -15.85108150     2992
```

Dengan cara yang sama, kita ulangi perhitungannya dengan mode *unshifted* (0 0 0):

Data untuk mode *unshifted*

```
# k-points  Shift Energi total (Ry) Grid non-ekuivalen
  1 0 -14.59239644     1
  ... ... ...      ...
  32  0 -15.85108141     897
```

Apabila kedua data tersebut kita plot, maka kita akan memperoleh grafik sebagai berikut:



Dari grafik tersebut kita bisa melihat bahwa energi total ketika kita menggunakan`k-points` sebanyak $4\times4\times4$ hanya berbeda < 2 meV/atom dengan energi total bila kita gunakan `k-points` sebanyak $32 \times 32 \times 32$. Untuk mode *shifted*, terlihat bahwa konvergensinya lebih cepat tercapai jika dibandingkan mode *unshifted*. Jadi, dalam kasus ini, kita sebaiknya menggunakan parameter `k-points` 4 4 4 1 1 1.

2.5 Struktur Setimbang

Pada bagian ini, kita akan belajar untuk menghitung struktur setimbang dari suatu sistem yang sederhana. Kali ini, mari kita fokus saja pada aturan berikut:

Diantara semua struktur yang mungkin, struktur setimbang dari suatu sistem (suhu $T = 0$, tekanan $P = 0$) dapat diperoleh dengan meminimalkan energi total dari sistem tersebut.

Energi total yang dimaksud ialah energi total hasil perhitungan dari DFT, seperti yang kita gunakan pada tutorial sebelumnya. Energi total ini mencakup seluruh *term* pada Hamiltonian elektron-inti, kecuali energi kinetik dari inti yang diabaikan karena kita menganggap inti-inti tersebut diam (tidak bergerak). Energi ini biasa disebut sebagai energi potensial permukaan.

2.5.1 Struktur Setimbang dari Molekul Diatomik

Mari kita menghitung keseimbangan struktur molekul Cl_2. Molekul Cl_2 hanya memiliki 2 atom, dan strukturnya hanya ditentukan oleh panjang ikatan Cl–Cl. Oleh karena itu, kita dapat menentukan struktur yang seimbang dengan menghitung energi total sebagai fungsi dari jarak Cl–Cl.

Pertama-tama, kita mencari *pseudopotential* yang cocok untuk atom Cl. Kita akan menggunakan *pseudopotential* LDA dengan label *pz* pada *file* .UPF-nya.

```
$ wget  http://www.quantum-espresso.org/upf_files/Cl.pz-bhs.UPF
```

Setelah itu, kita siapkan *file* input (`cl2.scf.in`) seperti berikut:

```
&CONTROL
  calculation = "scf",
  prefix = "Cl2",
  pseudo_dir = "./",
  outdir = "./"
/
&SYSTEM
  ibrav = 1,
  celldm(1) = 20.0,
  nat = 2,
  ntyp = 1,
  ecutwfc = 100,
/
&ELECTRONS
/
```

```
ATOMIC_SPECIES
  Cl   1.0   Cl.pz-bhs.UPF
ATOMIC_POSITIONS {bohr}
  Cl 0.00 0.00 0.00
  Cl 2.00 0.00 0.00
K_POINTS {gamma}
```

Dengan `ibrav = 1`, artinya kita memilih kotak simulasi berbentuk kubus sederhana dengan parameter kisi `celldm(1)`. Dalam hal ini, kita memilih sebuah kubus dengan sisi yang cukup besar, yakni 20 Bohr (1 Bohr = 0.529167 Å), untuk menghindari interaksi antar Cl_2. Ingat walaupun kita hanya mempunyai satu molekul di dalam kotak simulasi, tapi karena sistem akan direpetisi sebanyak tak-hingga, molekul Cl_2 akan "muncul" kembali dengan jarak 20 Bohr. Jadi, bila `celldm(1)` tidak cukup besar, maka kemungkinan masih ada interaksi antar molekul Cl_2.

Kita juga dapat melihat, bahwa di dalam file input ini, kata kunci `gamma` digunakan pada blok `K_POINTS`. Hal ini berarti kita akan mengambil zona Brillouin pada titik Γ, yaitu pada **k** = 0. Jumlah titik ini cukup karena kita hanya ingin meninjau molekul, bukan kristal periodik. Perhatikan juga bahwa kita meningkatkan energi *cutoff* dari fungsi gelombang sebesar 100 Ry. Nilai ini diperoleh secara terpisah dari uji konvergensi.

File input `cl2.scf.in` kemudian dijalankan dengan perintah berikut:

```
$ mpirun -np 4 pw.x < cl2.scf.in > cl2.scf.out
```

Setelah selesai, kita cek energi total dari sistem yang kita hitung:

```
$ grep "total energy" Cl2.out
     total energy =   -55.58060868 Ry
     total energy =   -55.80487514 Ry
     total energy =   -55.84085668 Ry
     total energy =   -55.84152500 Ry
     total energy =   -55.84160081 Ry
     total energy =   -55.84161854 Ry
     total energy =   -55.84162030 Ry
!    total energy =   -55.84162116 Ry
     The total energy is the sum of the following terms:
```

Di sini kita lihat bahwa energi mencapai minimum setalah 8 kali iterasi. Prosedur iterasi berhenti ketika perbedaan energi antara dua iterasi yang berurutan lebih kecil daripada `conv_thr = 1.0d-6`. Data terkait hal ini bisa kita temukan 1 baris di bawah "! total energy = -55.84162116 Ry " pada file output:

```
!    total energy =   -55.84162116 Ry
      estimated scf accuracy <   0.00000003 Ry
```

Apabila kita menginginkan akurasi yang lebih tinggi, kita dapat menentukan sendiri nilai `conv_thr` yang kita inginkan. Sebagai contoh,

```
&ELECTRONS
  conv_thr = 1.d-10
/
```

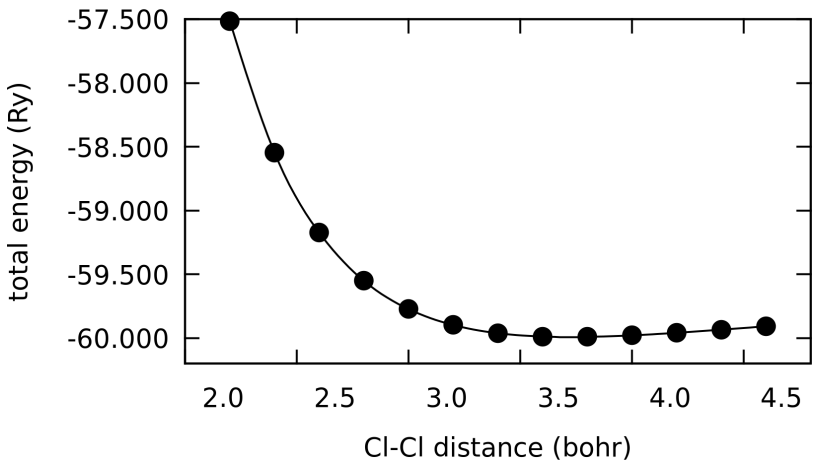
Selanjutnya kita ulangi perhitungan kita dengan variasi jarak antar atom Cl. Caranya, kita tinggal mengubah nilai X pada blok `ATOMIC_POSITIONS` dengan 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, 4.0, 4.2, 4.4, dan 4.6.

```
ATOMIC_POSITIONS {bohr}
  Cl  0.00  0.00  0.00
  Cl  X   0.00  0.00
```

Apabila sudah selesai, kita kumpulkan data energi total untuk setiap nilai X (jarak antar atom Cl), seperti berikut:

```
# X (Bohr)  Energi total (Ry)
  2.2    -57.51390370 Ry
  2.4    -58.54440032 Ry
  ...    ...
  4.6    -59.90658471 Ry
```

Dengan menggunakan interpolasi `spline` pada GNUPLOT, kita akan memperoleh grafik seperti berikut ini:



Bila kita memperbesar gambar di atas, kita akan memperoleh panjang ikatan Cl-Cl dengan energi minimum adalah 3.725 Bohr atau 1.97 Å. Nilai ini mendekati hasil pengukuran, yakni 1.99 Å, artinya hanya berbeda sekitar 1.5%.

2.5.2 Energi Ikat dari Molekul Diatomik

Setelah kita mendapatkan struktur setimbang dari molekul Cl_2, kini kita akan menghitung energi ikatnya. Energi ikat dari molekul tersebut dapat didefinisikan sebagai perbedaan energi dari 2

atom atom Cl sebelum dan sesudah berikatan.

$$E_{\text{ikat}} = E_{\text{Cl}_2} - 2E_{\text{Cl}}, \tag{2.1}$$

dimana, E_{Cl_2} merupakan energi total dari molekul Cl_2, dan E_{Cl} merupakan energi total dari atom Cl terisolasi.

Besar E_{Cl_2} dapat kita peroleh dengan melakukan perhitungan SCF dengan *file* input seperti `cl2.scf.in`. Kita hanya perlu mengubah sedikit bagian pada *file* input tersebut seperti berikut:

```
ATOMIC_POSITIONS {bohr}
  Cl   0.000 0.00  0.00
  Cl   3.725 0.00  0.00
```

Setelah perhitungan selesai, diperoleh energi total Cl_2 sebesar $E_{\text{Cl}_2} = -59.99059540$ Ry.

Selanjutnya, kita akan menghitung energi total untuk atom Cl terisolasi. Untuk kasus ini memang sedikit kompleks, sebab terdapat satu elektron tidak berpasangan pada kulit elektronik terluar, $3p^5$: $\boxed{\uparrow\downarrow}\boxed{\uparrow\downarrow}\boxed{\uparrow\ }$. Untuk memperhitungkan hal ini, kita melakukan kalkulasi ***spin-polarized*** dengan modifikasi *file* input sebelumnya.

```
&CONTROL
  calculation = "scf",
  prefix = "Cl2",
  pseudo_dir = "./",
  outdir = "./"
/
&SYSTEM
  ibrav = 1,
  celldm(1) = 20.0,
  nat = 1,
  ntyp = 1,
  ecutwfc = 100,
  nspin = 2,
  tot_magnetization = 1.0,
  occupations = 'smearing',
  degauss = 0.001
/
&ELECTRONS
/
ATOMIC_SPECIES
  Cl   1.0   Cl.pz-bhs.UPF
ATOMIC_POSITIONS {bohr}
  Cl 0.00 0.00 0.00
  Cl 2.00 0.00 0.00
K_POINTS {gamma}
```

Parameter `nspin = 2` ditetapkan agar program memperhitungkan spin elektron pada sistem. Parameter `tot_magnetization = 1.0` digunakan untuk memastikan 1 elektron tidak berpasangan. Setelah menjalankan `pw.x` pada file input yang telah dibuat, kita akan memperoleh $E_{\mathrm{Cl}} = -29.86386106$ Ry.

Dengan memasukkan kedua hasil yang kita peroleh ke dalam Persamaan 2.1, kita peroleh:

$$E_{\mathrm{ikat}} = 0.262875\,\mathrm{Ry} = 3.58\,\mathrm{eV}.$$

Jika dibandingkan dengan hasil eksperimen ($E_{\mathrm{ikat}} = 2.51$ eV),[7] nilai yang kita peroleh lebih besar sekitar 1 eV (43%). Dari sini kita bisa melihat bahwa ikatan interatomik akan terlalu kuat apabila kita menggunakan LDA.

2.5.3 Struktur Setimbang dari Kristal 3-Dimensi

Pada bagian ini, kita akan mencari struktur setimbang dari kristal silikon. Berikut ini adalah *file* input (`silicon-2.scf.in`) yang akan kita gunakan (perlu diperhatikan bahwa nilai parameter konvergensi yang dipakai sudah melalui proses pengecekan):

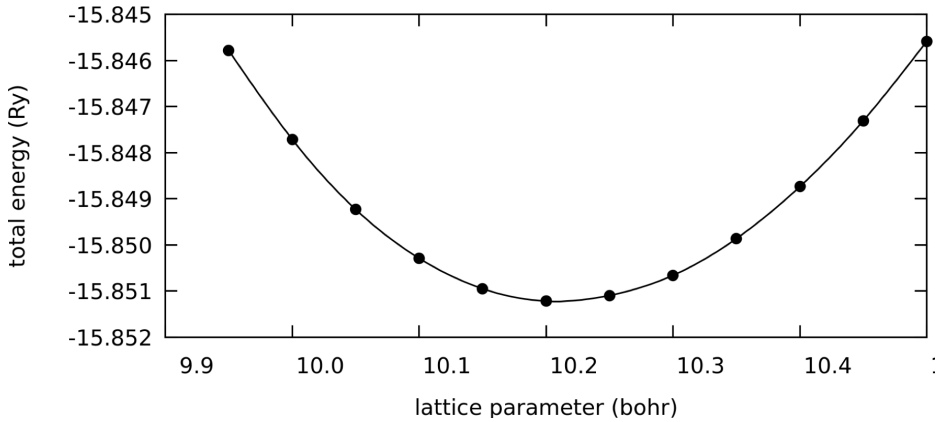
```
&CONTROL
  calculation = "scf",
  prefix = "silikon",
  pseudo_dir = "./",
  outdir = "./"
/
&SYSTEM
  ibrav = 2,
  celldm(1) = 9.95,
  nat = 2,
  ntyp = 1,
  ecutwfc = 25.0,
/
&ELECTRONS
/
ATOMIC_SPECIES
  Si  28.086 Si.pz-vbc.UPF
ATOMIC_POSITIONS {crystal}
  Si 0.00 0.00 0.00
  Si 0.25 0.25 0.25
K_POINTS {automatic}
  4 4 4 1 1 1
```

[7] https://en.wikipedia.org/wiki/Bond-dissociation_energy.

Pada umumnya, informasi mengenai struktur dari suatu kristal bisa kita peroleh dari pengukuran XRD (*X-Ray Diffraction*). Untuk kasus kristal silikon, struktur setimbangnya dapat ditentukan berdasarkan nilai konstanta kisinya (a). Dengan kata lain, kita perlu mencari nilai a dengan energi total yang minimum. Caranya dengan melakukan perhitungan untuk nilai `celldm(1)` yang berbeda: 9.95, 10.00, 10.05, 10.10, 10.15, 10.20, 10.25, 10.30, 10.35, 10.40, 10.45 dan 10.50. Setelah menjalankan `pw.x`, kita akan memperoleh energi total untuk setiap nilai a seperti berikut:

```
#   a (Bohr) Energi total (Ry)
    9.95     -15.84578279
    ...      ...
   10.50     -15.84558106
```

Apabila diplot, maka kita akan memperoleh grafik seperti berikut:



Di sini, kita juga melakukan interpolasi `spline` dengan menggunakan GNUPLOT (garis hitam). Dari grafik di atas, kita peroleh $a = 10.2078$ Bohr atau 5.4016 Åmerupakan parameter kisi dengan energi total minimum. Nilai ini mendekati nilai a hasil perhitungan,[8] yakni 5.43 Å(hanya berbeda 0.5%). Kembali kita dapat melihat bahwa hasil perhitungan menggunakan LDA menghasilkan nilai parameter kisi yang **lebih kecil (*underestimate*)** dari hasil pengukuran.

2.5.4 Energi Kohesi dari Kristal 3-Dimensi

Energi kohesi menunjukkan seberapa banyak energi yang dibutuhkan untuk menghancurkan ikatan-ikatan antar atom yang tersusun dalam kristal menjadi atom-atom yang terisolasi. Serupa dengan apa yang kita lakukan untuk menghitung energi ikat pada molekul Cl_2, kini kitapun akan menghitung perbedaan energi total kristal dan energi total atom terisolasi. Secara matematis,

[8] https://en.wikipedia.org/wiki/Monocrystalline_silicon

perumusan energi kohesi untuk tiap atom dapat ditulis sebagai berikut:

$$E_{\text{kohesi}} = \frac{1}{N}\left(E_{\text{Si_3D}} - NE_{\text{Si}_1}\right), \tag{2.2}$$

dimana, $E_{\text{Si_3D}}$ merupakan energi total dari kristal silikon 3D, E_{Si_1} merupakan energi total dari atom Si terisolasi, dan N merupakan jumlah atom Si di dalam sel unit kristal.

$E_{\text{Si_3D}}$ bisa kita peroleh dengan menjalankan `pw.x` pada *file* input `silicon-2.scf.in` dengan sedikit modifikasi pada blok `&SYSTEM`seperti berikut:

```
  ...
  celldm(1) = 10.2078,
  ...
```

Hasilnya, $E_{\text{Si_3D}} = -15.85122369$ Ry.

Selanjutnya, kita menghitung energi total atom Si terisolasi. Ingat bahwa kita perlu menggunakan sel unit yang besar ($\sim$ 20 Bohr) untuk menghindari interaksi antar atom. Sebagai tambahan, atom Si mempunyai 4 elektron valensi dengan konfigurasi $3s^2\ 3p^2$. Berdasarkan aturan Hund, spin elektronnya tersusun seperti ini, $3p^2$: $\boxed{\uparrow}\ \boxed{\uparrow}\ \boxed{}$. Dengan memperhatikan hal-hal tersebut, file input untuk atom Si terisolasi dapat ditulis sebagai berikut:

```
&CONTROL
  calculation = "scf",
  prefix = "silikon",
  pseudo_dir = "./",
  outdir = "./"
/
&SYSTEM
  ibrav = 1,
  celldm(1) = 20.0,
  nat = 1,
  ntyp = 1,
  ecutwfc = 25.0,
  nspin = 2,
  tot_magnetization = 2.0,
  occupations = 'smearing',
  degauss = 0.001,
/
&ELECTRONS
/
ATOMIC_SPECIES
  Si  28.086 Si.pz-vbc.UPF
ATOMIC_POSITIONS {crystal}
  Si 0.00 0.00 0.00
K_POINTS {gamma}
```

Untuk memastikan agar konfigurasi 2 elektron terluar tetap tidak berpasangan, maka kita tetapkan parameter `tot_magnetization = 2.0`. Setelah menjalankan `pw.x` pada *file* input tersebut, maka kita peroleh $E_{\mathrm{Si}_1} = -7.53721918$ Ry.

Dengan memasukkan nilai $E_{\mathrm{Si_3D}}$, E_{Si_1}, dan $N = 2$ ke dalam Persamaan 2.2, diperoleh $E_{\mathrm{kohesi}} = 0.38839$ Ry atau 5.28 eV. Jika dibandingkan dengan hasil pengukuran secara eksperimen (4.62 eV), nilai yang diperoleh dari perhitungan memiliki nilai yang lebih besar sekitar 14%. Jadi, kita bisa simpulkan bahwa penggunaan LDA akan menghasilkan nilai parameter kisi yang lebih kecil dan energi ikat / energi kohesi yang lebih tinggi dari eksperimen. Hal ini merupakan ciri khas dari LDA. Sebagai catatan tambahan, untuk GGA-PBE,[9] umumnya akan diperoleh parameter kisi yang sedikit lebih besar ($\sim 1\%$) dari eksperimen dan energi ikat / energi kohesi yang lebih rendah.

2.6 Visualisasi Struktur Kristal

Cukup sulit jika kita hanya membayangkan struktur kristal dari *file* input yang kita buat. Oleh karena itu, kita perlu menvisualisasikannya agar bisa memastikan bahwa *file* input yang kita buat sudah menunjukkan struktur yang benar. Ada beberapa *software* yang dapat kita gunakan untuk memvisualisasikan struktur kristal, diantaranya: XCRYSDEN, VESTA, dll.

2.6.1 XCrySDen

XCRYSDEN[10] - *(X-Window) Crystalline Structures and Densities* merupakan *software* gratis yang bisa kita gunakan untuk memvisualisasikan struktur kristal dan juga molekul, serta densitas elektron dari suatu sistem. XCRYSDEN dapat membaca *file* input atau output dari QE. Informasi lengkap mengenai *software* ini dapat dilihat pada tautan berikut: http://www.xcrysden.org.

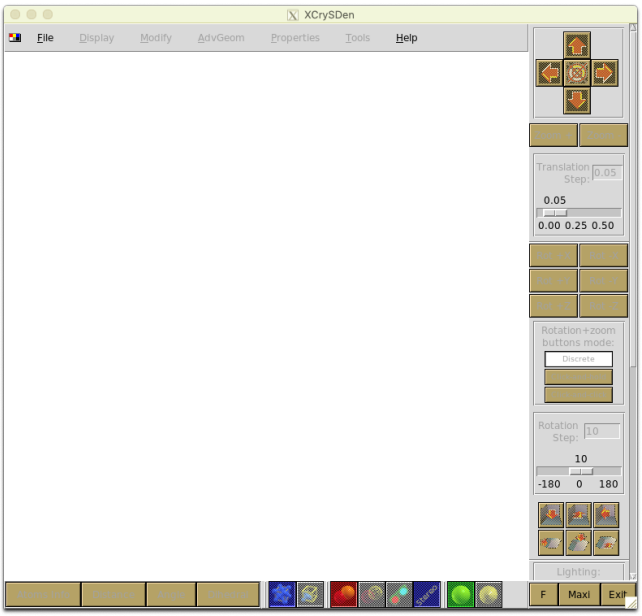
Kita persiapkan terlebih dahulu *file* input untuk sistem *graphene* (`graphene.scf.in`), kemudian kita visualisasikan dengan XCRYSDEN.

```
&CONTROL
  calculation = "scf",
  prefix = "graphene",
  pseudo_dir = "./",
  outdir = "./"
/
&SYSTEM
  ibrav = 4,
```

[9] Perdew, Burke, Ernzerhof, PRL **77**, 3865 (1996)
[10] A. Kokalk, J. Mo. Graphics Modelling **17**, 176 (1999)

```
  a = 2.4623,
  c = 10
  nat = 2,
  ntyp = 1,
  occupations = 'smearing',
  smearing = 'methfessel-paxton',
  degauss = 0.02,
  ecutwfc = 60,
  ecutrho = 720,
  nbnd = 8,
/
&ELECTRONS
/
ATOMIC_SPECIES
  C 12.0107 C.pbe-rrkjus.UPF
ATOMIC_POSITIONS {crystal}
  C  0.333333333  0.666666666  0.500000000
  C  0.666666666  0.333333333  0.500000000
K_POINTS {automatic}
  12 12 1 0 0 0
```

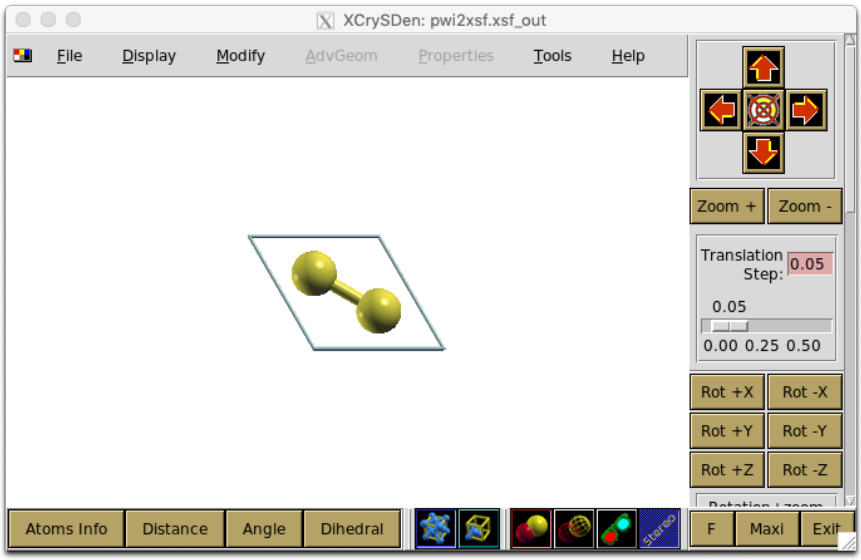
Kini, mari kita menjalankan software XCRYSDEN dan akan muncul jendela seperti ditunjukkan oleh Gambar 2.3.[11]



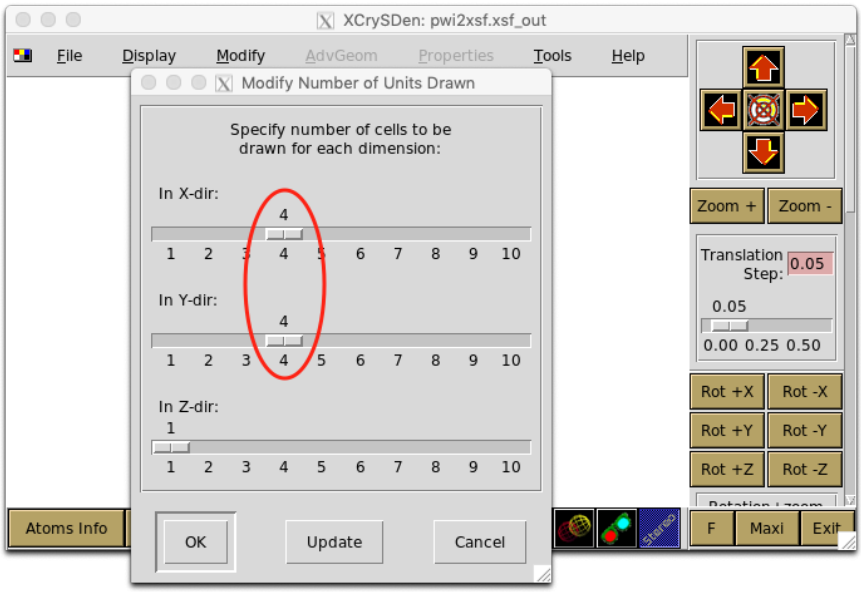
Gambar 2.3: Tampilan awal XCRYSDEN.

[11]Tampilan latar belakang yang sebelumnya berwarna hitam telah diubah menjadi putih.

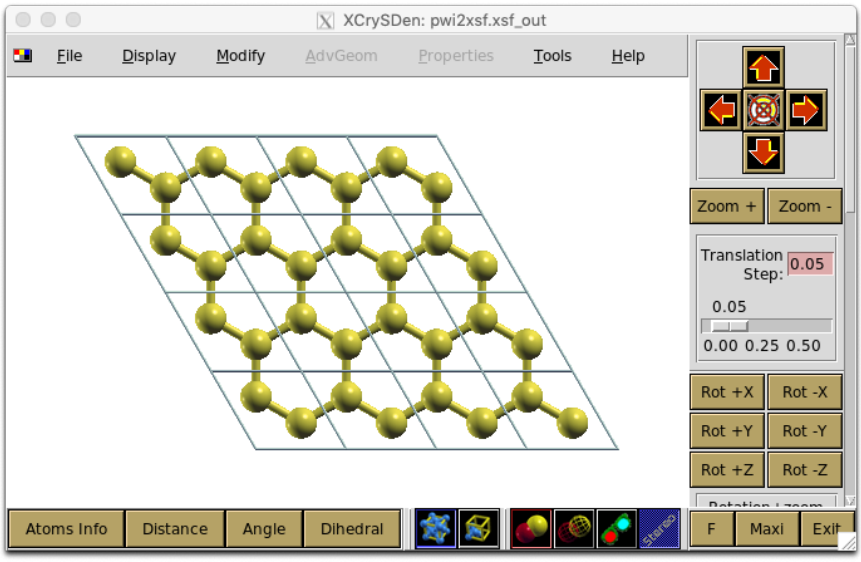
Untuk membuka file input , kita dapat memilih `File` → `Open PWscf ...` → `Open PWscf Input File`, dan menuju ke lokasi dimana file input disimpan dan tekan tombol `OK`. Selanjutnya akan muncul 2 jendela baru: untuk jendela yang berisi tentang rincian file input bisa kita tutup dengan menekan tombol `close`, untuk jendela kedua kita bisa memilih opsi `do not reduce dimensionallity` dan diakhir dengan menekan tombol `OK`. Hasilnya, akan muncul 2 atom C seperti gambar berikut:



Apabila kita hanya menampilkan hanya 1 sel unit saja, gambar visualisasinya tidak seperti graphene. Oleh karena itu, kita perlu melakukan repetisi untuk arah-x dan arah-y dengan cara memilih menu `Modify` → `Number of Units Drawn`. Kali ini, kita atur agar XCRYSDEN merepetisi struktur yang kita punya sebanyak $4\times$ untuk arah-x dan arah-y.



Hasilnya akan seperti berikut:



2.6.2 VESTA

VESTA[12] - *Visualization for Electronic and STructural Analysis* juga merupakan program visualisasi 3D untuk struktur kristal, data volumetrik seperti densitas elektron, dan morfologi kristal. VESTA juga dapat digunakan secara gratis untuk keperluan akademik, riset, dan kegiatan non-komersial lainnya. Informasi lengkap tentang *software* ini dapat dilihat melalui tautan ini: https://jp-minerals.org/vesta/en/. Berbeda dengan XCRYSDEN, VESTA tidak bisa membaca *file* input dan output QE secara langsung. Kita perlu mengubah *file* `.in` atau `.out` ke dalam format `.xsf` (*Xcrysden Structure File*) atau `.xyz`.

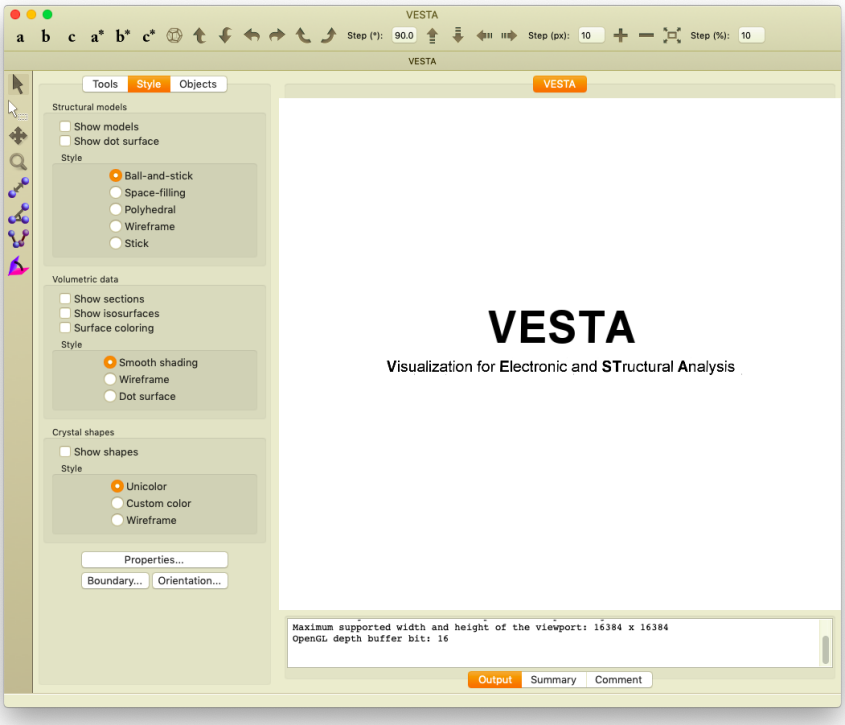
Kali ini kita akan menggunakan XCRYSDEN untuk membuat *file* `.xsf`. Caranya, buka file input QE yang akan diubah formatnya (misalnya `graphene.scf.in`) dengan XCRYSDEN. Kemudian, pilih `File` → `Save XSF Structure`. Ketikkan nama *file* (misalnya `graphene.xsf`) dan tekan tombol `Save`. Setelah itu, kita akan memperoleh *file* `graphene.xsf` yang isinya sebagai berikut:

```
INFO
nunit      1    1    1
unit   cell
celltype   convcell
shape   parapipedal
END_INFO
DIM-GROUP
```

[12] K. Momma and F. Izumi. J. Appl. Crystallogr. **44**, 1272 (2011)

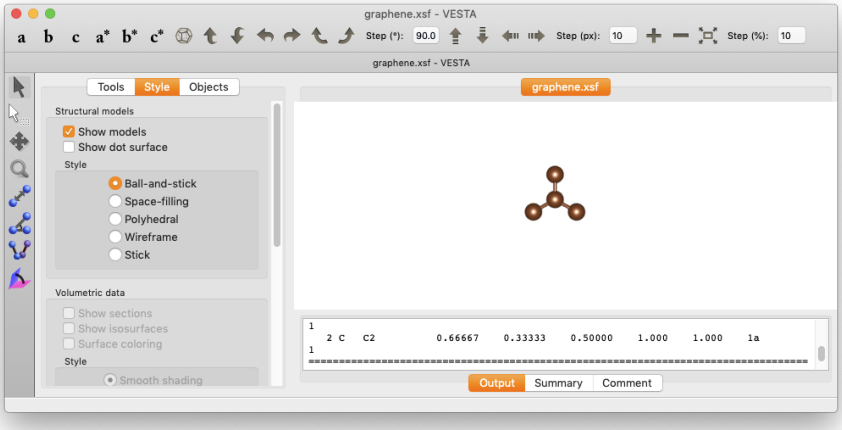
```
            3            9
PRIMVEC
     2.4623000000     0.0000000000     0.0000000000
    -1.2311500000     2.1324143520     0.0000000000
     0.0000000000     0.0000000000    10.0000000000
CONVVEC
     2.4623000000     0.0000000000     0.0000000000
    -1.2311500000     2.1324143520     0.0000000000
     0.0000000000     0.0000000000    10.0000000000
PRIMCOORD
            2            1
  6       0.0000000000      1.4216095660      5.0000000000
  6       1.2311499990      0.7108047830      5.0000000000
  .
  .
  .
  [dan seterusnya]
```

Jika kita menjalankan VESTA maka tampilan awalnya akan tampak seperti Gambar 2.4.

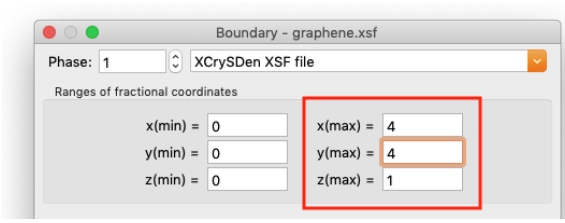


Gambar 2.4: Tampilan awal VESTA.

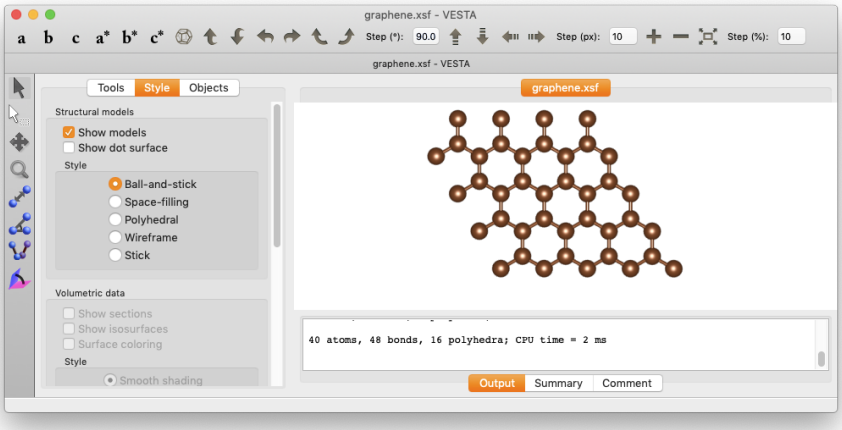
Kemudian pilih `FIle` → `Open` dan arahkan ke tempat dimana kita menyimpan *file* `graphene.xsf`, maka akan diperoleh gambar sebagai berikut: Di sini pun kita tidak bisa



melihat struktur *graphene* dengan jelas karena memang kita hanya menampilkan 1 sel unit saja. Untuk merepetisi sel unit $4\times$ ke arah-x dan arah-y, kita pilih `Objects` → `Boundary` dan tetapkan nilai `x(max) = 4` dan `y(max) = 4`.



Hasilnya,



2.7 Relaksasi Struktur

Pada bagian ini kita akan belajar untuk menemukan struktur paling stabil dari suatu sistem menggunakan metoda optimasi otomatis.

2.7.1 Optimasi Koordinat Atom

Pada Sub-BAB 2.5.1, kita menentukan struktur setimbang dari molekul Cl_2 dengan melakukan perhitungan SCF (`calculation = 'scf'`) untuk nilai jarak antar atom Cl yang bervariasi. Setelah itu data energi total sebagai fungsi dari jarak dikumpulkan, kemudian diinterpolasi, dan terakhir dicari jarak antar atom Cl dengan energi total minimum. Kini, kita tidak perlu melakukan perhitungan yang berulang-ulang seperti itu.

QE menyediakan perhitungan otomatis untuk mengoptimasi struktur suatu sistem, atau yang disering disebut relaksasi struktur. Salah satunya, kita dapat menggunakan mode `calculation = 'relax'` pada blok `&CONTROL`. Pada prakteknya, QE akan menghitung besar gaya (*force*) yang terjadi pada masing-masing atom/ion. Posisi atom kemudian akan diperbaharui guna meminimalisasi gaya tersebut. Akhirnya, struktur setimbang dari sistem tersebut akan diperoleh ketika:

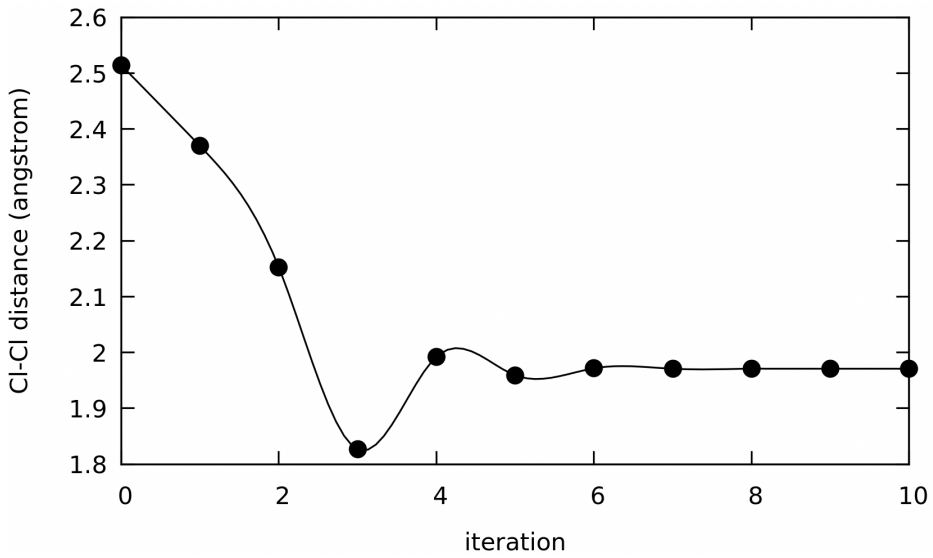
- Gaya untuk semua atom lebih kecil dari nilai *threshold* (`forc_conv_thr`) yang ditentukan sebelumnya (default: 1.0D-3), dan
- Perubahan energi total dari iterasi sebelumnya lebih kecil dari nilai *threshold* (`etot_conv_thr`) yang telah ditentukan (default: 1.0D-4).

Perhitungan dengan mode ini, hanya membutuhkan 1 *file* input. Mari kita kembali mencari struktur setimbang dari molekul Cl_2 dengan `calculation = 'relax'`.

```
&CONTROL
  calculation = "relax",
  prefix = "Cl2",
  pseudo_dir = "./",
  outdir = "./",
  forc_conv_thr = 1.d-5,
  etot_conv_thr = 1.d-8,
/
&SYSTEM
  ibrav = 1,
  celldm(1) = 20.0,
  nat = 1,
  ntyp = 1,
  ecutwfc = 100,
  nspin = 2,
  tot_magnetization = 1.0,
```

```
  occupations = 'smearing',
  degauss = 0.001
/
&ELECTRONS
/
&IONS
/
ATOMIC_SPECIES
  Cl   1.0   Cl.pz-bhs.UPF
ATOMIC_POSITIONS {bohr}
  Cl 0.00 0.00 0.00
  Cl 5.00 0.00 0.00
K_POINTS {gamma}
```

Berbeda dengan *file* input sebelumnya (`silicon-2.scf.in`), kini kita menambahkan parameter `forc_conv_hr = 1.d-5`, parameter `etot_conv_thr = 1.d-8`, dan blok `&IONS`. Setelah `pw.x` dijalankan, maka setelah beberapa iterasi kita akan memperoleh struktur setimbang dari molekul Cl_2. Berikut ini rekam jejak relaksasi strukturnya:



Dari grafik di atas kita bisa melihat bahwa relaksasi struktur Cl_2 membutuhkan 10 kali iterasi. Bila kita membuka *file* output dan mencari kata kunci `"final"`, maka kita akan memperoleh informasi sebagai berikut:

```
Final energy    =      -59.9905957145 Ry
...
Begin final coordinates
ATOMIC_POSITIONS (bohr)
  Cl   0.6377132745 0.0000000000 0.0000000000
  Cl   4.3622867255 0.0000000000 0.0000000000
End final coordinates
```

Informasi tersebut menunjukkan data struktur setimbang dari Cl_2 yang kita hitung. Jaraknya, $4.3622867255 - 0.6377132745 = 3.72457$ Bohr atau 1.97 Å. Nilai ini sesuai dengan nilai yang kita peroleh dari perhitungan struktur setimbang secara manual pada Sub-BAB 2.5.1.

2.7.2 Optimasi Koordinat Atom dan Sel Unit

Setelah sebelumnya kita telah belajar melakukan perhitungan optimasi struktur secara otomatis dengan membiarkan atom bergerak mencari posisi setimbangnya, sedangkan sel unit **tidak berubah**. Beberapa kasus, seperti optimasi struktur kristal, membutuhkan optimasi yang membiarkan atom bergerak mencari posisi setimbangnya sekaligus mengizinkan sel unit juga berubah. QE ternyata juga menyediakan mode ini, yakni `calculation = 'vc-relax'`. Kini, mari kita coba belajar menggunakan mode ini untuk menghitung struktur setimbang dari *graphene*.

Mari kita siapkan *file* input (`graphene.vc-relax.in`) seperti berikut:

```
&CONTROL
  calculation = "vc-relax",
  prefix = "graphene",
  pseudo_dir = "./",
  outdir = "./",
  forc_conv_thr = 1d-5,
/
&SYSTEM
  ibrav = 4,
  celldm(1) = 2.4,
  celldm(3) = 10.0,
  nat = 2,
  ntyp = 1,
  occupations = 'smearing',
  smearing = 'mp',
  degauss = 0.02,
  ecutwfc = 60,
  ecutrho = 720,
  nbnd = 8,
/
&ELECTRONS
  conv_thr = 1.0d-8,
  mixing_beta = 0.7,
/
&IONS
/
&CELL
  cell_dofree = '2Dxy',
/
```

```
ATOMIC_SPECIES
  Cl   1.0   C.pbe-rrkjus.UPF
ATOMIC_POSITIONS {crystal}
  Cl 0.33333 0.66667 0.00
  Cl 0.66667 0.33333 0.00
K_POINTS {automatic}
  12 12 1 1 1 1
```

Setelah `pw.x` dijalankan pada *file* input di atas, kita ketikan perintah berikut untuk mengetahui rekam jejak relaksasi strukturnya:

```
$ grep ! graphene.vc-relax.out
```

Maka akan tampil informasi berikut:

```
!    total energy              =      -22.78395481 Ry
!    total energy              =      -22.79047397 Ry
!    total energy              =      -22.79080063 Ry
!    total energy              =      -22.79080343 Ry
!    total energy              =      -22.79080739 Ry
```

Dari informasi di atas kita dapat mengetahui bahwa telah terjadi 5 kali iterasi ionik hingga diperoleh struktur setimbang dari *graphene*.

Dengan membuka *file* output dan mencari kata kunci `"final"`, kita akan memperoleh informasi:

```
Final enthalpy =      -22.7908034268 Ry
Begin final coordinates
     new unit-cell volume =    354.37900 a.u.^3 (    52.51355 Ang^3
   )

CELL_PARAMETERS (alat=  4.53534272)
   1.026027806  -0.000000000   0.000000000
  -0.513013903   0.888566145   0.000000000
   0.000000000   0.000000000   4.166666667

ATOMIC_POSITIONS (crystal)
C    0.333333333   0.666666666   0.500000000
C    0.666666666   0.333333333   0.500000000
End final coordinates
```

Dari informasi ini kita dapat menghitung nilai konstanta kisi untuk *graphene* yang telah kita hitung, yakni $4.53534272 \times 1.026027806 = 4.65$ Bohr atau 2.4607 Å. Nilai ini sangat mendekati nilai konstanta kisi hasil pengukuran,[13] yakni 2.46 Å.

[13] https://en.wikipedia.org/wiki/Graphene

2.8 Menghitung *Density of States* dan *Band Gap*

Pada bagian ini kita akan menghitung *density of states* atau DOS dan *band gap* dari kristal Si 3D. Perhitungan ini terdiri dari 4 tahap, yaitu:

- Perhitungan `scf`,
- Perhitungan `nscf`,
- Peritungan `dos`, dan
- Membuat grafik `dos`.

2.8.1 Perhitungan `scf`

Pada tahap pertama ini, kita siapkan input (`silicon.scf.in`) sebagai berikut:

```
&CONTROL
  calculation = 'scf',
  prefix = 'silicon'
  outdir = './'
  pseudo_dir = './'
/
&SYSTEM
  ibrav=2,
  celldm(1) =10.410909236,
  nat=2,
  ntyp=1,
  ecutwfc=50,
/
&electrons
  conv_thr=1e-8
/
ATOMIC_SPECIES
  Si  28.0855  Si.pbe-n-rrkjus_psl.1.0.0.UPF

ATOMIC_POSITIONS (alat)
  Si 0.00 0.00 0.00
  Si 0.25 0.25 0.25
K_POINTS (automatic)
  8 8 8 0 0 0
```

Ketikkan perintah di bawah ini untuk menghitung *file* input tersebut.

```
$ mpirun -np 4 pw.x < silicon.scf.in > silicon.scf.out
```

Pada dasarnya, pada tahap ini kita akan menghitung *wavefunction* atau *charge density* yang konvergen untuk sistem tersebut.

2.8.2 Perhitungan nscf

Pada tahap ini kita tidak lagi melakukan perhitungan *self-consistent* (`calculation = 'nscf'`), sebab kita telah memiliki *charge density* yang konvergen dari perhitungan sebelumnya. Di sini, kita hanya meningkatkan jumlah k-point. Perlu diingat pula bahwa untuk perhitungan DOS, pada tahap ini kita sebaiknya menggunakan `occupations = 'tetrahedra'`. Dan apabila nantinya kita akan melanjutkan perhitungan kita untuk menghitung sifat termoelektrik dari suatu material, kita sebaiknya menambahkan paramter `verbosity = 'high'` agar penulisan data pada *file* output menjadi lebih lengkap.

Berikut ini *file* input (`silicon.nscf.in`) untuk perhitungan nscf:

```
&CONTROL
  calculation = 'nscf',
  prefix = 'silicon'
  outdir = './'
  pseudo_dir = './'
/
&SYSTEM
  ibrav=2,
  celldm(1) =10.410909236,
  nat=2,
  ntyp=1,
  ecutwfc=50,
  occupations='tetrahedra'
/
&electrons
  conv_thr=1e-8
/
ATOMIC_SPECIES
  Si  28.0855  Si.pbe-n-rrkjus_psl.1.0.0.UPF

ATOMIC_POSITIONS (alat)
  Si 0.00 0.00 0.00
  Si 0.25 0.25 0.25
K_POINTS (automatic)
  12 12 12 0 0 0
```

Sama seperti biasanya, kita juga menggunakan `pw.x` untuk menjalankan perhitungan `nscf` berdasarkan *file* input yang telah kita buat.

```
$ mpirun -np 4 pw.x < silicon.nscf.in > silicon.nscf.out
```

2.8.3 Perhitungan dos

Tahapan ini merupakan tahapan inti dari perhitungan dos. Kita perlu menyiapkan *file* input (silicon.dos.in) yang cukup berbeda dengan biasanya.

```
&DOS
    prefix='si'
    outdir='./outdir/'
    fildos='si.dos.dat'
    emin=-9.0
    emax=16.0
/
```

Seperti yang dapat kita lihat, *file* input tersebut hanya terdiri dari satu blok saja, yakni blok &DOS. Parameter prefix, namanya harus sama dengan prefix pada *file* input untuk perhitungan scf dan nscf; parameter outdir menunjukkan dimana kita akan meletakkan *file* output hasil perhitungan dos ini; parameter fildos merupakan parameter yang kita gunakan untuk menentukan nama *file* output; dan dua parameter terakhir menunjukkan nilai rentang energi yang akan dihitung DOSnya (dari energi emin sampai emax).

Untuk melakukan perhitungan dos ini, kita tidak menggunakan pw.x tetapi dos.x seperti pada perintah berikut:

```
$ mpirun -np 4 dos.x < silicon.dos.in > silicon.dos.out
```

Setelah selesai kita akan memperoleh *file* silicon.dos.dat, yang berisi data sebagai berikut:

```
#  E (eV)   dos(E)     Int dos(E) EFermi =    6.133 eV
  -9.000  0.0000E+00  0.0000E+00
  -8.990  0.0000E+00  0.0000E+00
  -8.980  0.0000E+00  0.0000E+00
  -8.970  0.0000E+00  0.0000E+00
  ...
  15.940  0.0000E+00  0.1600E+02
  15.950  0.0000E+00  0.1600E+02
  15.960  0.0000E+00  0.1600E+02
  15.970  0.0000E+00  0.1600E+02
  15.980  0.0000E+00  0.1600E+02
  15.990  0.0000E+00  0.1600E+02
  16.000  0.0000E+00  0.1600E+02
```

2.8.4 Membuat Grafik `dos`

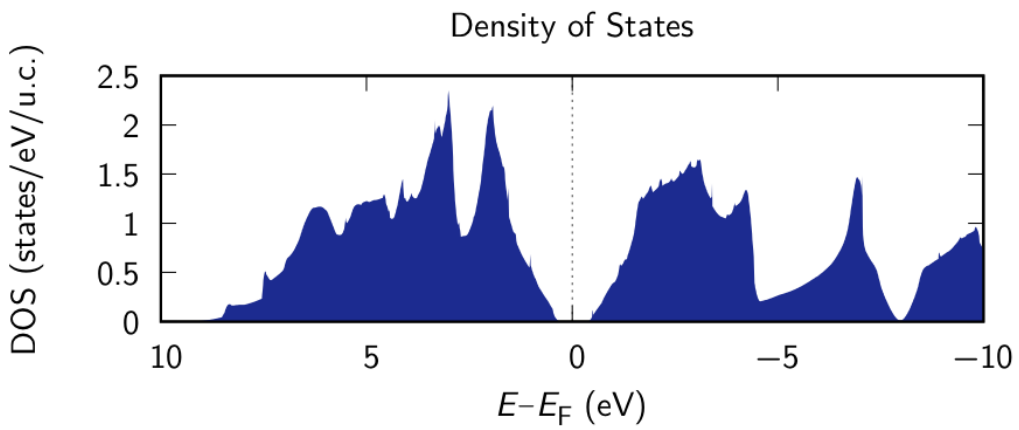
Tahap terakhir ialah memvisualisasikan `silicon.dos.dat` menjadi grafik DOS. Kita bisa menggunakan `software` apapun, tapi pada tutorial ini kita akan menggunakan GNUPLOT.[14] Mari kita buat *file* GNUPLOT (`plot.silicon.gnu`) yang berisi perintah-perintah untuk membuat grafik sesuai dengan yang kita inginkan.

```
e_Fermi    = 6.133
xmax = -10.00
xmin =  10.00
set size 0.6,0.4
set border lw 1
set xr[xmin:xmax]
set ylabel "DOS  (states/eV/u.c.)"
set xlabel "E-E_F (eV)"
set title "Density of States"
set yzeroaxis lc rgb "black"
plot "si.dos.dat" u ($1-e_Fermi):2 w filledcurves x=0 \
  lw 1.5 lc rgb "#0A2793" notitle
```

Kita bisa menjalankan *script* GNUPLOT tersebut dengan mengetikkan perintah berikut ini:

```
$ gnuplot plot.silicon.gnu
```

Hasilnya seperti gambar berikut:



Seperti yang terlihat pada grafik di atas, kita telah memposisikan energi Fermi di titik nol sumbu-x. Hal ini bisa dilakukan dengan menggeser nilai energi sebesar nilai energi Ferminya ($E - E_F$). Setelah berhasil memvisualisasikan data `dos`, saatnya kita menghitung *band gap* dari kristal

[14] Penjelasan mengenai penggunaan GNUPLOT dapat dilihat di BAB 5.

silikon ini. Nilai *band gap* ini dapat kita peroleh dengan menghitung rentang energi dengan `DOS=0` di sekitar energi Fermi. Kita bisa membuka *file* `si.dos.dat` dan melihat nilai DOS di sekitar energi Fermi ($E_F = 6.133$ eV).

```
...
5.730  0.3756E-02  0.8000E+01
5.740  0.1775E-02  0.8000E+01
5.750  0.5282E-03  0.8000E+01
5.760  0.1468E-04  0.8000E+01 <--
5.770  0.0000E+00  0.8000E+01
5.780  0.0000E+00  0.8000E+01
...
6.400  0.0000E+00  0.8000E+01
6.410  0.0000E+00  0.8000E+01
6.420  0.2293E-03  0.8000E+01 <--
6.430  0.9380E-03  0.8000E+01
6.440  0.2126E-02  0.8000E+01
...
```

Dari data di atas kita bisa menetapkan nilai pita valensi maksimum berada pada energi 5.76 eV dan pita konduksi minimum pada energi 6.42 eV. Oleh karena itu, kita peroleh *band gap* untuk kristal Si 3D sebesar $6.42 - 5.76 = 0.66$ eV. Nilai ini cukup jauh berbeda dengan *band gap* dari hasil pengukuran,[15] yakni 1.12 eV. Sudah menjadi rahasia umum bahwa DFT seringkali menghasilkan nilai *band gap* yang lebih kecil dari eksperimen. Untuk itu, perlu koreksi tambahan atau trik khusus untuk menyelesaikan permasalahan ini.

2.9 Menghitung *Band Structure* dan Visualisasi *Wavefunctions*

Pada bagian ini kita akan belajar untuk menghitung *band structure* untuk material semikonduktor dan metalik, serta memvisualisasikan *wavefunctions*.

2.9.1 *Band Structure* Kristal Silikon

Kini, kita akan menggunakan kristal silikon sebagai studi kasus sekaligus representasi dari material semikonduktor. Untuk menghitung *band structure*, ada beberapa tahap yang perlu dilakukan, yaitu:

- Perhitungan `scf`,
- Perhitungan `bands`,

[15] https://www.rp-photonics.com/band_gap.html

- Ekstraksi data `bands`, dan
- Membuat grafik *band structure*.

Perhitungan `scf`

Pada dasarnya, tahap ini sama dengan apa yang kita lakukan pada Sub-BAB 2.8.1, yaitu tahap perhitungan `scf` untuk menghitung DOS. Mari kita membuat *file* input `si.scf.in` seperti berikut:

```
&CONTROL
  calculation = "scf"
  prefix = "silicon"
  pseudo_dir = "./"
  outdir = "./"
/
&SYSTEM
  ibrav = 2
  celldm(1) = 10.2078
  nat = 2
  ntyp = 1
  ecutwfc = 25.0
/
&ELECTRONS
/
ATOMIC_SPECIES
  Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS {crystal}
  Si 0.00 0.00 0.00
  Si 0.25 0.25 0.25
K_POINTS {automatic}
  4 4 4 1 1 1
```

Setelah itu, kita eksekusi *file* input tersebut dengan program `pw.x` dengan perintah berikut:

```
$ mpirun -np 4 pw.x < si.scf.in > si.scf.out
```

Perhitungan `bands`

Selanjutnya, kita lanjut ke tahap perhitungan `bands`. Kali ini kita menggunakan mode `calculation = 'bands'` dan juga menentukan k-point dengan lebih spesifik. Berikut *file* input (`si.nscf-bands.in`) untuk menghitung `nscf`:

```
&CONTROL
  calculation = "bands"
```

```
  prefix = "silicon"
  pseudo_dir = "./"
  outdir = "./"
/
&SYSTEM
  ibrav = 2
  celldm(1) = 10.2078
  nat = 2
  ntyp = 1
  ecutwfc = 25.0
  nbnd = 8
/
&ELECTRONS
/
ATOMIC_SPECIES
  Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS {crystal}
  Si 0.00 0.00 0.00
  Si 0.25 0.25 0.25
K_POINTS {tpiba_b}
  3
  0.5 0.5 0.5 20
  0.0 0.0 0.0 20
  1.0 0.0 0.0 1
```

Seperti yang dapat kita lihat pada *file* input `si.nscf-bands.in`, selain menggunakan mode `calculation = 'bands'`, kita juga menambahkan parameter `nbnd = 8`. Hal ini berguna agar kita dapat melihat 4 pita valensi dan 4 pita konduksi terbawah. Kita juga menentukan secara spesifik `path` yang kita ingin kaji pada zona Brillouin. Pada kasus ini, kita memilih *path* dari titik $L = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})\frac{2\pi}{a}$ ke titik $\Gamma = (0, 0, 0)\frac{2\pi}{a}$ sebanyak 20 titik, kemudian dari $\Gamma = (0, 0, 0)\frac{2\pi}{a}$ ke $X = (1, 0, 0)\frac{2\pi}{a}$ juga sebanyak 20 titik.

Selanjutnya, *file* input `si.nscf-bands.in` kita eksekusi dengan menjalankan program `pw.x` dengan perintah berikut:

```
$ mpirun -np 4 pw.x < si.nscf-bands.in > si.nscf-bands.out
```

Ekstraksi Data `bands`

Selanjutnya, kita akan mengekstrak informasi *bands* dari hasil perhitungan sebelumnya. Untuk itu, kita perlu mempersiapkan *file* input (`si.bands.in`) seperti berikut:

```
&BANDS
  prefix  = "silicon"
  outdir = "./"
```

```
  filband = "bands"
```

Seperti yang bisa kita lihat, kita hanya memerlukan satu blok (&BANDS) pada *file* input ini. Untuk parameter prefix, kita harus menggunakan nilai yang sama dengan prefix pada *file* input untuk perhitungan scf dan bands. Parameter outdir menunjukkan posisi dimana *file* output akan diletakkan. Parameter filband merupakan nama untuk *file-file* output yang akan dihasilkan pada perhitungan ini.

Setelah *file* input dibuat, kita perlu mengeksekusinya dengan menjalankan program bands.x menggunakan perintah berikut:

```
$ mpirun -np 4 bands.x < si.bands.in > si.bands.out
```

Apabila proses perhitungannya telah selesai, kita akan memperoleh beberapa *file* output, salah satunya silicon.gnu. *File* ini berisi informasi tentang *bands* yang tersusun sedemikian rupa sehingga kita akan mudah memvisualisasikannya dengan GNUPLOT.

Membuat grafik *band structure*

Untuk membuat grafik *band structure* dari *file* output silicon.gnu, kita dapat menyiapkan terlebih dahulu *script* GNUPLOT seperti (plot.bands.gnu) berikut:

```
e_Fermi = 6.2106
unset key
set ylabel "Energy (eV)"
set xtics ("L" 0, "G" 0.866, "X" 1.866)
set xlabel "k-point path [2pi/a]"
plot for [i=2:9] "bands.gnu" using 1:(column(i)-e_Fermi) \
   w l lc 3 lw 2 smooth csplines, \
   for [i=2:9]"" using 1:(column(i)-e   zero) w p pt 7 lc 0 ps 0.5
```

Sebagai catatan, apabila kita menghitung material semikonduktor, kita tidak menggunakan parameter occupations. Dengan begitu *file* output si.scf.out tidak akan berisi informasi energi Fermi secara eksplisit (tidak kita temukan energi Fermi dengan kata kunci "Fermi". Oleh karena itu, yang bisa kita lakukan ialah dengan mencari informasi mengenai nilai energi untuk pita valensi maksimum dan pita konduksi minimum. Caranya, alih-alih menggunakan kata kunci "Fermi", kita menggunakan kata kunci highest.

```
$ grep highest si.scf.out
```

Maka akan muncul informasi berikut:

```
    highest occupied, lowest unoccupied level (ev):    X    Y
```

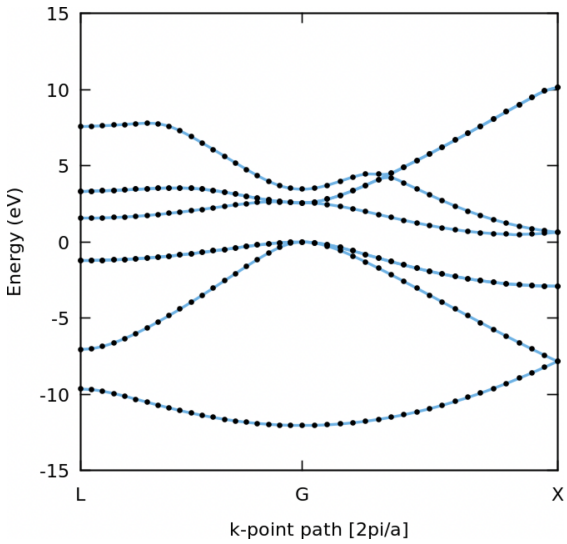
Nilai X merupakan nilai energi pita valensi maksimum $E_{\mathrm{V}}^{\max}$ dan Y merupakan nilai energi pita konduksi minimum $E_{\mathrm{C}}^{\min}$. Dengan begitu, nilai energi Fermi dapat diperoleh dengan rumus berikut:

$$E_{\mathrm{Fermi}} = \frac{1}{2}\left(E_{\mathrm{val}}^{\max} + E_{\mathrm{C}}^{\min}\right)$$

Kini, kita eksekusi *script* GNUPLOT yang telah kita buat dengan perintah berikut:

```
$ gnuplot silicon.gnu
```

Hasilnya seperti gambar berikut:



2.9.2 Visualisasi *Wavefunctions* Kristal Silikon

Setelah menghitung *band structure*, kita bisa juga memvisualisasikan *wavefunction* untuk setiap nilai eigen yang kita pilih. Untuk melakukan hal itu, kita membutuhkan program `pp.x`, yang seharusnya sudah tersedia bersama dengan program-program yang lain, seperti: `pw.x`, `dos.x`, dan `bands.x`. Program ini akan membaca *file* output ketika kita menjalankan program `pw.x` dan menuliskan data ke dalam format yang sesuai untuk *software* visualisasi standar. Kali ini, kita akan memvisualisasikan *wavefunction* dari kristal silikon yang sebelumnya kita hitung.

Berikut *file* input (`si.pp.in`) yang perlu kita siapkan:

```
&INPUTPP
  prefix = "silicon"
  outdir = "./",
  filplot = "wavefc"
  plot_num = 7
  lsign = .true.
```

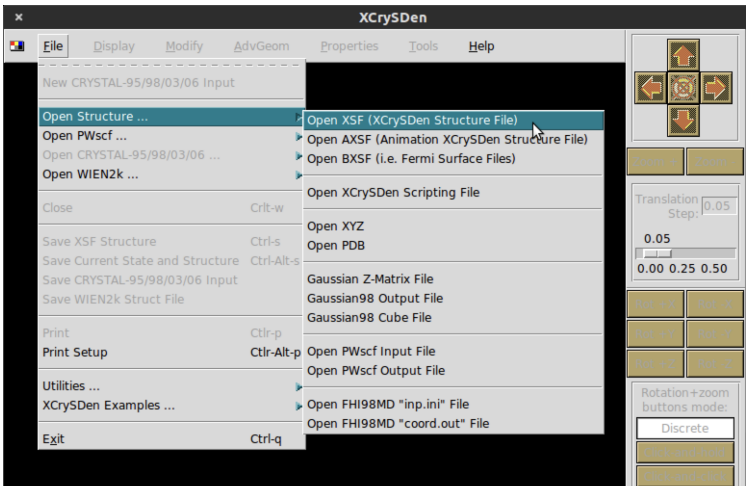
```
  kpoint = 21
  kband = 4
/
&PLOT
  iflag = 3
  output_format = 5
  fileout = "silicon.xsf"
/
```

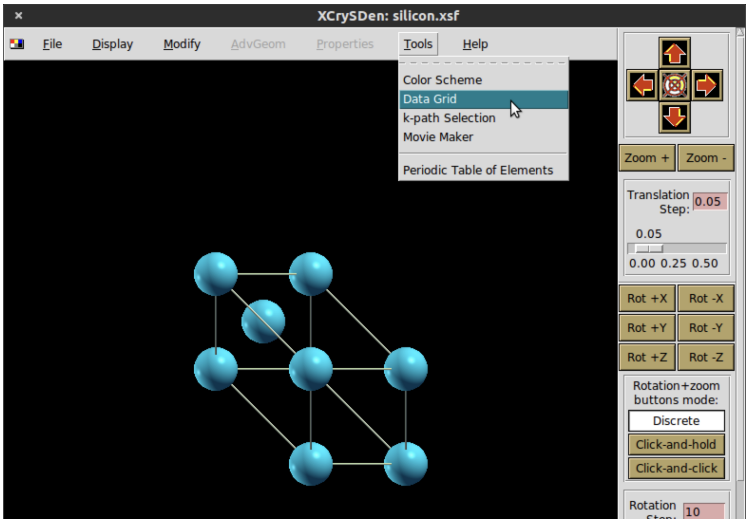
Di sini kita perlu menuliskan dua blok, `&INPUTPP` dan `&PLOT`. Ada beberapa parameter yang penting untuk diperhatikan. Parameter `plot_num = 7` menunjukkan bahwa parameter yang akan kita visualisasi ialah $|\Psi|^2$. Parameter `lsign = .true.` digunakan untuk membedakan tanda (positif atau negatif) dari *wavefunction*. Parameter `kpoint = 21` dan `kband = 4` untuk menunjukkan bahwa kita memilih titik ke-21 dan pita ke-4 untuk divisualisasikan. Ini merupakan posisi pita valensi maksimum pada titik Γ. Parameter `iflag = 3` dan `output_format = 5` digunakan agar output yang dihasilkan ditulis dalam format XCRYSDEN. Informasi lengkap mengenai program `pp.x` dapat dilihat di http://www.quantum-espresso.org/Doc/INPUT_PP.html

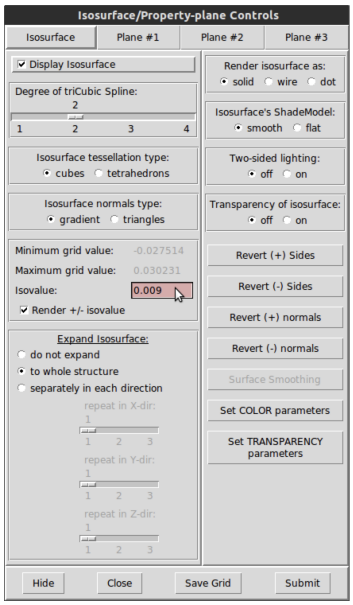
File input (`si.pp.in`) kemudian dieksekusi dengan dua perintah berikut:

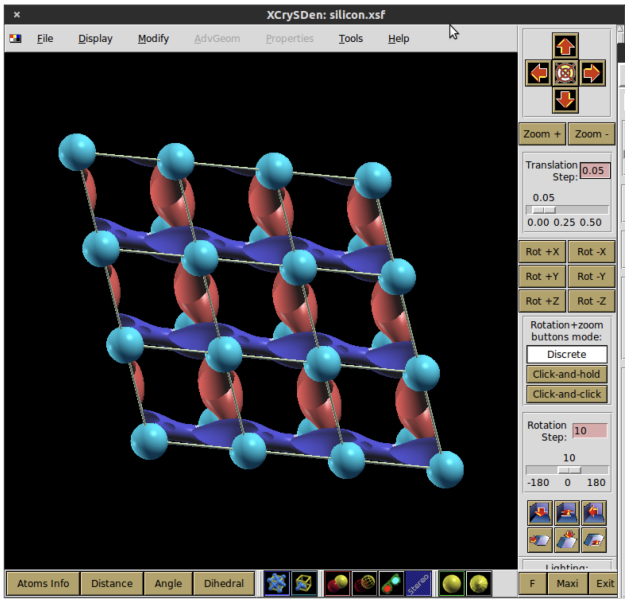
```
$ mpirun -np 4 pw.x < si.nscf-bands.in > si.nscf-bands.out
$ mpirun -np 4 pp.x < si.pp.in > si.pp.out
```

Setelah selesai kita akan memperoleh *file* output `silicon.xsf`. Selanjutnya, kita membutuhkan XCRYSDEN untuk memvisualisasikan data tersebut. Langkah-langkah yang dilakukan ditunjukkan oleh gambar-gambar berikut ini:









Pada contoh kasus ini, kita bisa melihat bahwa elektron-elektron di pita valensi maksimum terkonsentrasi di ikatan antar atom silikon, sebagaimana yang juga diperoleh melalui perhitungan menggunakan metoda *tight-binding*.

2.9.3 *Band Structure* Kristal Tembaga

Setelah menghitung *band structure* untuk material semikonduktor, sekarang kita akan menghitung *band structure* untuk material metalik dengan mengambil kristal tembaga sebagai contoh kasus.

Tahapan perhitungannya sama seperti pada saat kita menghitung *band structure* untuk kristal silikon. Hanya terdapat beberapa perbedaan di bagian parameter inputnya saja.

Perhitungan scf

Berikut *file* input (cu.scf.in) yang perlu kita siapkan untuk tahapan pertama:

```
&CONTROL
  calculation = "scf",
  prefix = "copper",
  pseudo_dir = "./",
  outdir = "./"
/
&SYSTEM
  ibrav = 2,
  celldm(1) = 6.678,
  nat = 1,
  ntyp = 1,
  ecutwfc = 40,
  ecutrho = 300,
  occupations = "smearing",
  smearing = "mp",
  degauss = 0.01,
/
&ELECTRONS
/
ATOMIC_SPECIES
  Cu 1.0 Cu.pz-n-van_ak.UPF
ATOMIC_POSITIONS {alat}
  Cu 0.00 0.00 0.00
K_POINTS {automatic}
  8 8 8 1 1 1
```

Secara umum, *file* input di atas tidak berbeda dengan *file* input untuk perhitungan scf sebelumnya. Perbedaan mendasar hanya pada tiga parameter tambahan yang ada pada blok &SYSTEM, yakni parameter occupations, smearing, dan degauss. **Ketiga parameter ini perlu kita tambahkan ketika menghitung material metalik.**

Seperti biasa, kita eksekusi input tersebut dengan menjalankan program pw.x:

```
$ mpirun -np 4 pw.x < cu.scf.in > cu.scf.out
```

Karena kita menggunakan parameter occupations, smearing, dan degauss, maka kita akan memperoleh data energi Fermi secara eksplisit. Dengan menggunakan kata kunci "Fermi" pada *file* output, kita akan memperoleh informasi sebagai berikut:

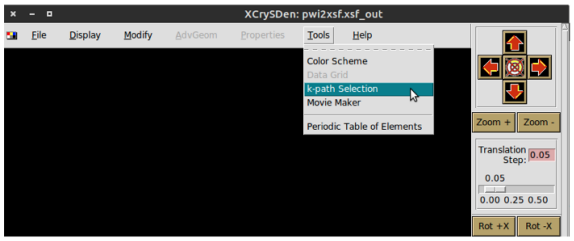
```
  the Fermi energy is     13.3640 eV
```

Perhitungan bands

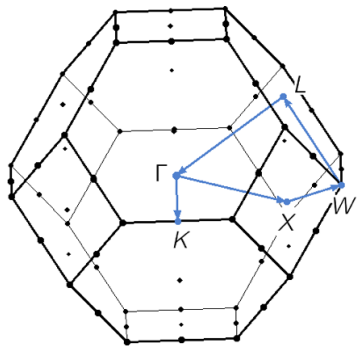
Selanjutnya, kita melakukan perhitungan *bands* dan menyiapkan *file* input sebagai berikut:

```
&CONTROL
  calculation = "bands",
  prefix = "copper",
  pseudo_dir = "./",
  outdir = "./"
/
&SYSTEM
  ibrav = 2,
  celldm(1) = 6.678,
  nat = 1,
  ntyp = 1,
  ecutwfc = 40,
  ecutrho = 300,
  occupations = "smearing",
  smearing = "mp",
  degauss = 0.01,
  nbnd = 8,
/
&ELECTRONS
/
ATOMIC_SPECIES
  Cu 1.0 Cu.pz-n-van_ak.UPF
ATOMIC_POSITIONS {alat}
  Cu 0.00 0.00 0.00
K_POINTS {tpiba_b}
  6
  0.00 0.00 0.00 50 ! G
  0.00 1.00 0.00 50 ! X
  0.50 1.00 0.00 50 ! W
  0.50 0.50 0.50 50 ! L
  0.00 0.00 0.00 50 ! G
  0.75 0.75 0.00 1  ! K
```

Di sini, kita akan menghitung *wavefunction* untuk 8 pita (`nbnd` =8), dengan *path* dari $\Gamma \rightarrow X \rightarrow W \rightarrow L \rightarrow \Gamma \rightarrow K$ dengan 50 titik untuk masing-masing *path*. Agar lebih mudah, kita bisa memanfaatkan `XCrySDen` untuk menentukan *path* pada zona Brillouin. Caranya ditunjukkan oleh gambar berikut:



Zona Brillouin Kristal Tembaga



Ekstraksi Data `bands`

Selanjutnya, kita akan mengekstrak informasi *bands* dari hasil perhitungan sebelumnya. Untuk itu, kita perlu mempersiapkan *file* input (`cu.bands.in`) seperti berikut:

```
&BANDS
  prefix  = "copper"
  outdir = "./"
  filband = "bands"
```

Sama seperti ekstraksi data *bands* kristal Silikon, kita pun perlu mengeksekusi *file* input di atas dengan perintah berikut:

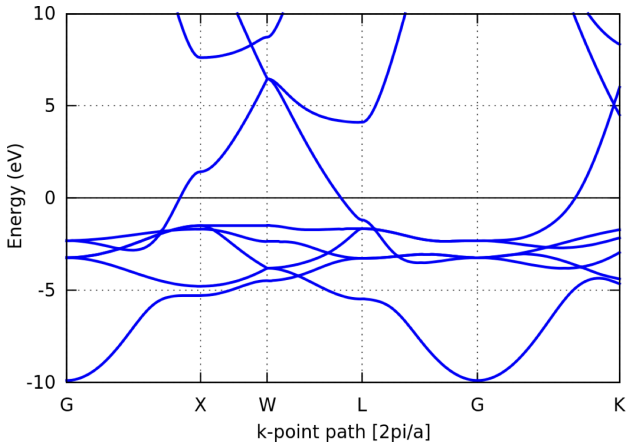
```
$ mpirun -np 4 bands.x < cu.bands.in > cu.bands.out
```

Membuat grafik *band structure*

Dari perhitungan sebelumnya, kita akan mendapatkan *file* output `copper.gnu`. Apabila kita visualisasikan dengan GNUPLOT, kita akan memperoleh gambar sebagai berikut:

Referensi lebih lanjut:

- User Guide: https://www.quantum-espresso.org/Doc/user_guide.pdf
- Hands On: https://www.paradim.org/sites/default/files/2019-05/Feliciano%27s%20Summer%20School%20Handout%202018%20%281%29.pdf


10
5
0
-5
-10
Energy (eV)
G
X
W
L
G
K
k-point path [2pi/a]

3 | BoltzTraP2

OLEH: EDI SUPRAYOGA

BOLTZTRAP2 - ***Boltz**mann **Tra**nsport **P**roperties* versi ke-2 merupakan *software* untuk menghitung koefisien transpor semi-klasik. *Software* ini menggunakan *mesh* pita energi yang diperoleh dari *software* berbasis DFT, seperti: QUANTUM ESPRESSO, Abinit, Siesta, VASP, dan Wien2K. Salah satu kegunaan BOLTZTRAP2 yang paling umum adalah untuk menghitung koefisien transpor elektronik (seperti koefisien Seebeck, konduktivitas listrik, dan konduktivitas termal) sebagai fungsi dari temperatur dan potensial kimia. Selain itu, *software* ini juga bisa melakukan visualisasi untuk *Fermi surface* secara 3D.

BOLTZTRAP2 ditulis dengan bahasa pemrograman PYTHON dan sebagian kecil C++. Untuk menjalankan BOLTZTRAP2, kita memastikan bahwa komputer yang kita gunakan sudah terpasang `Python` versi 3.5 atau versi yang lebih terbaru. Selain itu, *library* Python Numpy, Scipy, matplotlib, spglib, dan ASE sebaiknya juga sudah disiapkan. Semua paket program tersebut dapat dengan mudah diperoleh melalui *Python Package Index* (PyPI) menggunakan *tools* seperti `'pip'`. Selain itu, kita juga dapat memperolehnya melalui *bundle* distribusi Python seperti `'Anaconda'` dan sejumlah distribusi Linux. Jika kita menggunakan `'pip'` untuk menginstall BOLTZTRAP2, semua *library* tersebut sudah terinstal secara otomatis.

3.1 Instalasi BoltzTrap2

Apabila kita sudah memastikan bahwa program `pip` dan `Python` versi 3.5 ke atas sudah terinstal, kita dapat melakukan instalasi BOLTZTRAP2 dengan menggunakan perintah berikut pada jendela `terminal` kita.

```
$ pip3 install BoltzTraP2
```

Perintah tersebut akan mengunduh dan menginstal secara otomatis semua *library* `Python` yang dibutuhkan oleh `BoltzTraP2`. Setelah berhasil, maka *file binary* `'btp2'` akan terpasang pada komputer kita. File inilah yang kita gunakan untuk mengeksekusi perhitungan koefisien transpor

elektronik menggunakan `BoltzTraP2`. Keterangan lebih lanjut dapat kita peroleh melalui laman resmi `BoltzTraP2`.[1]

3.2 Menghitung Koefisien Transpor Elektronik GaAs

Modul praktis ini akan memandu kita untuk mengetahui cara penggunaan beberapa fitur utama BoltzTraP2 melalui contoh yang spesifik. Dalam hal ini, material yang akan dikaji adalah *Galium Arsenite* (GaAs) yang dapat diunduh pada tautan `http://quasi.id/download.html`. Kita dapat mengunduh *file* tersebut dan meletakkannya di direktori mana pun di komputer kita masing-masing. Untuk keperluan tutorial ini, kita asumsikan bahwa *file* yang telah diunduh diletakkan pada `'Download/GaAs'`.

Perhitungan koefisien transpor elektronik suatu material didasari oleh persamaan transpor Boltzmann (*Boltzmann transport equation* atau BTE) dengan pendekatan waktu relaksasi (τ) yang konstan (*constant relaxation time approximation* atau CRTA) dan *rigid-band approximation* (RBA).

Secara singkat, kita bisa mendefinisikan suatu kernel koefisien transpor $\mathcal{L}$ sebagai berikut:

$$\mathcal{L}_i = \int_{-\infty}^{\infty} \mathcal{T}(E)\,(E-\mu)^i \left(-\frac{\partial f}{\partial E}\right) dE, \quad (i = 1, 2, 3)$$

dengan

$$\mathcal{T}(E) = v_x^2(E)\,\tau(E)\,\mathcal{D}(E) \tag{3.1}$$

dimana,

$\mathcal{T}$	: Fungsi distribusi transpor.	f	: Fungsi distribusi Fermi-Dirac.
v_x	: *Carrier velocity*.	μ	: Potensial kimia.
τ	: Waktu relaksasi.	E	: Energi.
$\mathcal{D}$	: *Density of State*.		

Dengan kernel ini, selanjutnya kita dapat menghitung koefisien Seebeck (S), konduktivitas listrik (σ), dan konduktivitas termal (κ_e) dengan menggunakan persamaan-persamaan berikut:

$$S = \frac{1}{qT}\frac{\mathcal{L}_1}{\mathcal{L}_0}, \quad \sigma = \mathcal{L}_0, \quad \kappa_e = \frac{1}{q^2T}\left[\frac{\mathcal{L}_1^2}{\mathcal{L}_0} - \mathcal{L}_2\right],$$

dimana, q merupakan muatan elektron, dan T adalah temperatur.

[1] https://www.imc.tuwien.ac.at/index.php?id=21094

3.2.1 Menghitung DOS dengan Quantum Espresso

Seperti yang ditunjukkan pada Persamaan 3.1, kita perlu menghitung DOS untuk dapat menghitung fungsi distribusi transpor, yang kemudian digunakan untuk menghitung kernel transpor dan terakhir menghitung koefisien-koefisien transpor. DOS dapat kita hitung dengan menggunakan QE seperti yang telah kita pelajari pada sub-BAB 2.8. Pertama, mari kita siapkan *file* input (`GaAs.scf.in`) untuk perhitungan `scf` seperti berikut:

```
&CONTROL
  calculation = 'scf'
  prefix = 'GaAs'
  pseudo_dir = './'
  wf_collect = .true.
/
&SYSTEM
  ibrav = 2
  celldm(1) = 10.6867
  nat = 2
  ntyp = 2
  ecutwfc = 60
  ecutrho = 244
/
&ELECTRONS
  mixing_mode = 'plain'
  mixing_beta = 0.7
  conv_thr = 1.0d-8
/
ATOMIC_SPECIES
  Ga 69.723 Ga.pbe-dn-kjpaw_psl.1.0.0.UPF
  As 74.921595 As.pbe-n-kjpaw_psl.1.0.0.UPF
ATOMIC_POSITIONS
  Ga 0.00 0.00 0.00
  As 0.25 0.25 0.25
K_POINTS {automatic}
  8 8 8 0 0 0
```

Seperti biasa, kita eksekusi *file* input tersebut dengan menjalankan program `pw.x` melalui perintah berikut:

```
$ mpirun -np 4 pw.x < GaAs.scf.in > GaAs.scf.out
```

Selanjutnya, kita lanjut ke perhitungan `nscf` dengan terlebih dahulu menyiapkan *file* input (`GaAs.nscf.out`) seperti berikut:

```
&CONTROL
  calculation = 'nscf'
  verbosity = 'high'
```

```
  prefix = 'GaAs'
  pseudo_dir = './'
  wf_collect = .true.
/
&SYSTEM
  ibrav = 2
  celldm(1) = 10.6867
  nat = 2
  ntyp = 2
  ecutwfc = 60
  ecutrho = 244
  occupations = 'smearing'
  degauss = 0.005
/
&ELECTRONS
  mixing_mode = 'plain'
  mixing_beta = 0.7
  conv_thr = 1.0d-8
/
ATOMIC_SPECIES
  Ga 69.723 Ga.pbe-dn-kjpaw_psl.1.0.0.UPF
  As 74.921595 As.pbe-n-kjpaw_psl.1.0.0.UPF
ATOMIC_POSITIONS
  Ga 0.00 0.00 0.00
  As 0.25 0.25 0.25
K_POINTS {automatic}
  32 32 32 0 0 0
```

Seperti yang bisa kita lihat, kita menggunakan mode `calculation = 'nscf'`, sebab kali ini kita tidak melakukan perhitungan *self-consistent* lagi. Pada tahap ini, kita semacam melakukan "interpolasi" dengan memperbanyak jumlah k-point. Kali ini, kita menggunakan k-point sebanyak $32 \times 32 \times 32$. Hal lain yang perlu diperhatikan ialah penggunaan parameter `occupations`, kali ini kita menggunakan `occupations = 'smearing'`. Namun, akan lebih baik juga apabila kita menggunakan `occupations = 'tetrahedra'`.

Setelah *file* input disiapkan, selanjutnya kita eksekusi dengan program `pw.x` seperti berikut:

```
$ mpirun -np 4 pw.x < GaAs.nscf.in > GaAs.nscf.out
```

File output yang dihasilkan (`GaAs.nscf.out`) inilah yang akan kita pakai ketika menggunakan BoltzTrap2.

3.2.2 Memperoleh Nilai Energi Fermi

Selanjutnya, kita akan mengambil data mengenai energi Fermi. Data ini bisa kita peroleh dari *file* output `GaAs.nscf.out` dengan mengetikkan perintah berikut:

```
$ grep Fermi GaAs.nscf.out
```

Setelah itu, maka kita akan memperoleh informasi: $E_{\mathrm{F}} = 6.9789$ eV.

3.2.3 Mengkonversi Format *File* Output QE

Pada tahap ini, kita memerlukan program tambahan yang ditulis dalam bahasa Python, yakni `qe2boltz2.py`. Program ini berguna untuk mengubah format *file* output dari QE sedemikian rupa sehingga sesuai dengan format yang diperlukan oleh BoltzTraP2. Program `qe2boltz2.py` dapat diunduh melalui tautan berikut: http://quasi.id/download.html. Untuk dapat menjalankan program tersebut, kita perlu melakukan langkah-langkah berikut:

- Salin `qe2boltz2.py` ke direktori `bin`, tempat dimana program-program QE (`pw.x`, `dos.x`, dll.) tersimpan.

```
$ cp qe2boltz2.py [path_QE]/bin/.
```

- Masuk ke dalam direktori `bin` tersebut.

```
$ cd [path_QE]/bin/.
```

- Apabila kita menggunakan komputer berbasis UNIX (seperti: Linux atau MacOS), kita perlu memastikan tidak ada kesalahan format penulisan `Python` dari Windows.

```
$ dos2unix qe2boltz2.py
```

- Kita pastikan pula untuk menambahkan mode `execute` pada `qe2boltz2.py` agar program tersebut bisa dieksekusi.

```
$ chmod +x qe2boltz2.py
```

Untuk mengkonversi file output dari QE, kita gunakan perintah berikut:

```
$ qe2boltz2.py GaAs pw 6.0789 0
```

Suku pertama (`qe2boltz2.py`) untuk memanggil program `qe2boltz2.py` diikuti oleh nama `prefix` yang kita gunakan dalam perhitungan `scf` dan `nscf`. Suku selanjutnya menyatakan program yang kita gunakan dalam QE, yakni `pw.x`. Jadi, kita tuliskan `pw` pada suku ketiga. Suku keempat merupakan nilai energi Fermi. Suku terakhir menyatakan jumlah pita energi yang **tidak** akan kita sertakan dalam perhitungan (*exclude*). Pada kasus ini, kita menetapkan angka 0, artinya tidak ada pita energi yang tidak kita sertakan.

3.2.4 Melakukan Interpolasi Pita Energi

Pada tahap ini, kita akan melakukan interpolasi pada pita energi GaAs. BoltzTrP2 akan membaca data setiap pita energi sebagai polinomial trigonometri. Pada tahap ini, proses komputasi numerik akan memproses data mengenai struktur elektronik GaAs untuk memperoleh nilai koefisien dari setiap suku polinomial tersebut. Berikut perintah yang kita ketikkan untuk melakukan interpolasi pita energi:

```
$ btp2 interpolate -m 1 -e -0.5 ../qe_GaAs
```

Dua suku pertama (`btp2 interpolate`) merupakan perintah interpolasi untuk menghitung koefisien-koefisien polinomial. Opsi `-m 1` menyatakan seberapa halus interpolasi yang kita akan lakukan. Pada kasus ini, kita menginginkan agar program `btp2` melakukan *sampling* pada tiap 1 k-point (tidak tereduksi) untuk masing-masing titik yang tertulis di dalam *file* input. Opsi `-e -0.5` menyatakan energi minimum yang akan diperhitungkan. Disini, kita menentukan -0.5 eV sebagai batas energi minimum. Suku terakhir menyatakan posisi (*path*) dimana kita meletakkan *file* hasil perhitungan QE.

Setelah perintah selesai dijalankan, kita akan memperoleh *file* output `interpolation.bt2`. Apabila kita ingin menentukan nama *file* output secara spesifik, kita dapat menggunakan opsi `-o` diikuti dengan nama *file* yang kita inginkan. *File* output dengan ekstensi `.bt2` ini sebenarnya merupakan *file* JSON terkompresi dalam format `xz` yang dapat divisualisasikan dan dimanipulasi menggunakan beberapa *tools* dan *library* Python. Untuk melihat isi dari *file* tersebut kita dapat mengetikkan perintah berikut:

```
$ xzcat interpolation.bt2 | more
```

Sebagian besar pengguna hanya akan berinteraksi melalui interface `btp2` dan dapat dengan aman memperlakukan file `.bt2` sebagai *black box*.

3.2.5 Melakukan Integrasi Hasil Interpolasi

Hasil interpolasi pita energi yang tertulis dalam *file* `interpolation.bt2` selanjutnya akan digunakan untuk proses integrasi, yakni untuk menghitung transpor elektronik sistem. Dalam hal ini, kita menggunakan model *uniform-relaxation-time*. Interpolasi ini akan kita lakukan untuk rentang temperatur $50 \leq T \leq 800$ K dengan interval 50 K. Perintah yang perlu kita ketikkan ialah sebagai berikut:

```
$ btp2 integrate interpolation.bt2 50:800:50
```

Dua suku pertama (`btp2 integrate`) merupakan perintah integrasi untuk menghitung koefisien-koefisien transpor. Suku berikutnya menyatakan nama *file* output yang akan menyimpan data-data hasil integrasi. Suku terakhir menyatakan rentang temperatur dan

interval integrasinya. Perhitungan ini memerlukan waktu yang cukup lama, bergantung pada kemampuan komputer yang digunakan.

Setelah selesai, kita akan memperoleh *file* output dengan dua format yang berbeda, yaitu:

- *File* JSON baru yang terkompresi dengan nama yang sama dengan *file* input tetapi memiliki ekstensi `.btj`. *File* ini lebih nyaman digunakan untuk pemrosesan otomatis pada `btp2`.
- Kumpulan *file* teks yang mengikuti format yang sama seperti `BoltzTraP` versi terdahulu, yakni file dengan ekstensi '`.condtens`', '`.halltens`', dan '`.trace`'. *File-file* tersebut dapat dengan mudah divisualisasikan dengan hampir semua program visualisasi data apa pun. Baris pertama di setiap *file* tersebut merupakan *header* yang menjelaskan besaran dan unit pada setiap kolom.

3.2.6 Visualisasi Koefisien Seebeck

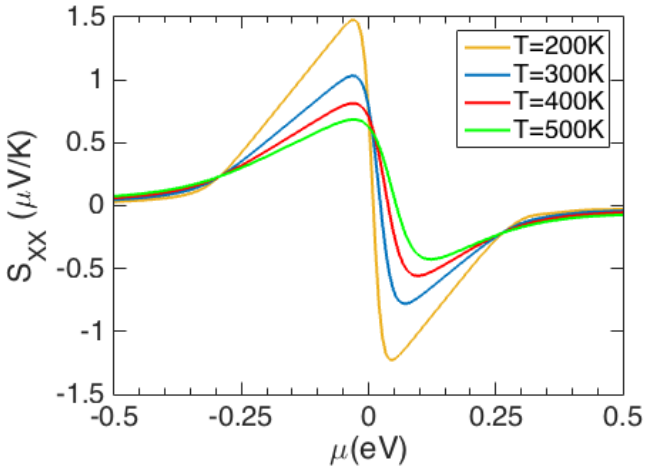
Tahap terakhir ialah memvisualisasikan data yang telah kita peroleh. Disini, kita hanya akan memvisualisasikan koefisien Seebeck sebagai contoh. Berikut ini perintah yang perlu kita ketikkan:

```
$ btp2 plot -u -c  '["xx", "zz"]' -s 50 interpolation.btj S
```

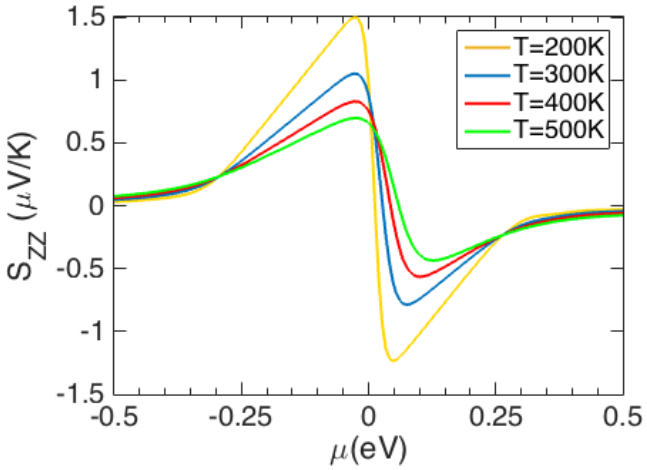
Dua suku pertama (`btp2 plot`) merupakan perintah *plotting* untuk koefisien yang akan kita visualisasikan. Opsi `-u` digunakan agar koefisien transpor diplot terhadap potensial kimia μ. Apabila kita ingin memvisualisaikan koefisien transpor sebagai fungsi temperatur, maka kita harus menggunakan opsi `-T`. Suku `-c '["xx", "zz"]'` digunakan untuk menentukan komponen mana yang ingin ditinjau. Disini, kita memilih komponen xx dan zz. Perlu diperhatikan bahwa tanda kutip perlu digunakan agar argumen string tetap dalam satu kesatuan walaupun terdapat spasi di dalamnya. Suku `-s 50` berguna untuk memerintahkan program `btp2` melakukan *sub-sampling* sebanyak 50 titik pada nilai temperatur yang lain. Dua suku terakhir (`interpolation.btj S`) masing-masing merupakan nama *file* yang berisi data koefisien-koefisien transpor dan koefiesien yang akan ditinjau (S: koefisien Seebeck). Setelah perintah dijalankan, kita akan memperoleh grafik seperti yang ditunjukkan oleh Gambar 3.1 dan 3.2.

Sejauh ini, Kita telah berhasil menghitung koefisien transpor elektronik pada GaAs menggunakan QE dan BoltzTraP2. Selain fitur yang kita pelajari ini, `BoltzTraP2` juga dapat digunakan untuk membuat visualisasi pita energi dan permukaan Fermi. Informasi lengkap mengenai penggunaan fitur-fitur BoltzTraP2 yang lain bisa dipelajari langsung melalui resmi `BoltzTraP2`[2]. Jika ingin mengetahui versi BoltzTraP sebelumnya, kami telah membuat video tutorial yang dapat dipelajari di https://youtu.be/X8O3tXwTp9Y.

[2]https://gitlab.com/sousaw/BoltzTraP2/-/wikis/tutorial



Gambar 3.1: Koefisien Seebeck (komponen xx).



Gambar 3.2: Koefisien Seebeck (komponen zz).

4 | Wannier90

OLEH: M. YUSRUL HANNA

WANNIER90 merupakan salah satu *platform post-processing open source* yang digunakan untuk membangkitkan fungsi `Wannier` yang terlokalisir maksimal (*Maximally-localized Wannier functions*). Dengan menggunakan program ini, struktur pita (*bandstructures*) dapat diwakilkan dengan model *tight-binding* yang memudahkan untuk melakukan perhitungan dengan k-point yang lebih halus. Sekarang, beberapa *software* simulasi material berbasis DFT, seperti: QUANTUM ESPRESSO, VASP, AbInit, OpenMX, Elk, Wien2k, Siesta dan FLEUR sudah terintegrasi dengan paket program WANNIER90. Pada studi kasus ini kita menggunakan QE untuk bagian perhitungan DFT-nya, dengan *graphene* sebagai objek material yang akan dikaji. Dengan material tersebut, kita akan mencoba melakukan perhitungan *post-processing* dengan program WANNIER90 untuk mengekstrak Hamiltonian untuk model *tight-binding*.

4.1 Instalasi Wannier90

Sebelum melakukan perhitungan *post-processing* dengan menggunakan program WANNIER90, pertama kita harus menginstalnya terlebih dahulu. *Installer* WANNIER90 bisa diunduh melalui tautan berikut: http://www.wannier.org/download/. Berikut langkah-langkah instalasi WANNIER90:

- Lakukan ekstraksi pada *installer* WANNIER90.

```
$ tar -xzvf wannier90-x.x.x.tar.gz
$ cd wannier90-x.x.x
```

- Lakukan konfigurasi agar instalasi dapat menyesuaikan *environment* komputer.
  Bagi pengguna *compiler* `gfortran`:

```
$ cp ./config/make.inc.gfort ./make.inc
```

  Bagi pengguna *compiler* `intel`:

```
$ cp ./config/make.inc.ifort ./make.inc
```

- Untuk mendefinisikan kompilasi paralel pada perhitungan `postw90.x`, maka masukkan **COMMS = mpi** pada `make.inc`.
- Lakukan instalasi.

```
$ make
```

- Tambahkan *path* instalasi WANNIER90 ke dalam `/.bashrc`

```
$ export PATH: wannier90-x.x.x/ :$PATH
$ source ~/.bashrc
```

Apabila instalasi dan penambahan *path* telah berhasil, maka perintah `wannier90.x` sudah bisa kita akses melalui `terminal`.

4.2 Konstruksi Hamiltonian Model *Tight-binding* pada *Graphene*

Untuk studi kasus kali ini, kita akan membuat fungsi gelombang `Wannier` yang terlokalisir maksimal untuk orbital sp^2 pada pita valensi *graphene* dan orbital π di sekitar *Fermi level*. Langkah pertama, kita perlu menyiapkan *file* input untuk perhitungan `scf` dan `nscf`. Berikut *file* input (`graphene.scf.in`) untuk perhitungan `scf`:

```
&CONTROL
  calculation = 'scf'
  prefix = 'graphene'
  pseudo_dir = './'
  outdir = './'
  wf_collect = .true.
  tstress = .true.
  tprnfor = .true.
/
&SYSTEM
  ibrav = 4,
  celldm(1) = 4.602,
  celldm(3) = 4,
  nat = 2,
  ntyp = 1,
  ecutwfc = 30.0,
  ecutrho = 150.0,
  occupations  = 'smearing'
  smearing = 'mp'
  degauss = 0.01
/
&ELECTRONS
  mixing_beta = 0.7
```

```
  conv_thr = 1.0d-8
/
ATOMIC_SPECIES
  C  12.0107  C.pz-van_ak.UPF
ATOMIC_POSITIONS {alat}
  C 0.000 0.000 0.000
  C 0.000 0.577 0.000
K_POINTS {automatic}
  12 12 1 0 0 0
```

Untuk perhitungan `nscf`, kita perlu menggunakan cukup banyak k-point. Disini, kita menggunakan 144 k-point, dan berikut *file* input (`graphene.nscf.in`) untuk perhitungan `nscf`:

```
&CONTROL
  calculation = 'nscf'
  prefix = 'graphene'
  pseudo_dir = './'
  outdir = './'
  wf_collect = .true.
  tstress = .true.
  tprnfor = .true.
/
&SYSTEM
  ibrav = 4,
  celldm(1) = 4.602,
  celldm(3) = 4,
  nat = 2,
  ntyp = 1,
  ecutwfc = 30.0,
  ecutrho = 150.0,
  occupations  = 'smearing'
  smearing = 'mp'
  degauss = 0.01
/
&ELECTRONS
  mixing_beta = 0.7
  conv_thr = 1.0d-8
/
ATOMIC_SPECIES {alat}
  C  12.0107  C.pz-van_ak.UPF
ATOMIC_POSITIONS
  C 0.000 0.000 0.000
  C 0.000 0.577 0.000
K_POINTS {crystal}
144
  0.00000000  0.00000000  0.00000000  6.944444e-03
```

```
  0.00000000  0.08333333  0.00000000  6.944444e-03
  0.00000000  0.16666667  0.00000000  6.944444e-03
  0.00000000  0.25000000  0.00000000  6.944444e-03
  0.00000000  0.33333333  0.00000000  6.944444e-03
  0.00000000  0.41666667  0.00000000  6.944444e-03
  0.00000000  0.50000000  0.00000000  6.944444e-03
  0.00000000  0.58333333  0.00000000  6.944444e-03
  0.00000000  0.66666667  0.00000000  6.944444e-03
  0.00000000  0.75000000  0.00000000  6.944444e-03
  0.00000000  0.83333333  0.00000000  6.944444e-03
  0.00000000  0.91666667  0.00000000  6.944444e-03
  0.08333333  0.00000000  0.00000000  6.944444e-03
  0.08333333  0.08333333  0.00000000  6.944444e-03
  0.08333333  0.16666667  0.00000000  6.944444e-03
  0.08333333  0.25000000  0.00000000  6.944444e-03
  0.08333333  0.33333333  0.00000000  6.944444e-03
  0.08333333  0.41666667  0.00000000  6.944444e-03
  0.08333333  0.50000000  0.00000000  6.944444e-03
  0.08333333  0.58333333  0.00000000  6.944444e-03
  0.08333333  0.66666667  0.00000000  6.944444e-03
  0.08333333  0.75000000  0.00000000  6.944444e-03
  0.08333333  0.83333333  0.00000000  6.944444e-03
  0.08333333  0.91666667  0.00000000  6.944444e-03
  0.16666667  0.00000000  0.00000000  6.944444e-03
  0.16666667  0.08333333  0.00000000  6.944444e-03
  0.16666667  0.16666667  0.00000000  6.944444e-03
  0.16666667  0.25000000  0.00000000  6.944444e-03
  0.16666667  0.33333333  0.00000000  6.944444e-03
  0.16666667  0.41666667  0.00000000  6.944444e-03
  0.16666667  0.50000000  0.00000000  6.944444e-03
  0.16666667  0.58333333  0.00000000  6.944444e-03
  0.16666667  0.66666667  0.00000000  6.944444e-03
  0.16666667  0.75000000  0.00000000  6.944444e-03
  0.16666667  0.83333333  0.00000000  6.944444e-03
  0.16666667  0.91666667  0.00000000  6.944444e-03
  0.25000000  0.00000000  0.00000000  6.944444e-03
  0.25000000  0.08333333  0.00000000  6.944444e-03
  0.25000000  0.16666667  0.00000000  6.944444e-03
  0.25000000  0.25000000  0.00000000  6.944444e-03
  0.25000000  0.33333333  0.00000000  6.944444e-03
  0.25000000  0.41666667  0.00000000  6.944444e-03
  0.25000000  0.50000000  0.00000000  6.944444e-03
  0.25000000  0.58333333  0.00000000  6.944444e-03
  0.25000000  0.66666667  0.00000000  6.944444e-03
  0.25000000  0.75000000  0.00000000  6.944444e-03
  0.25000000  0.83333333  0.00000000  6.944444e-03
```

```
0.25000000  0.91666667  0.00000000  6.944444e-03
0.33333333  0.00000000  0.00000000  6.944444e-03
0.33333333  0.08333333  0.00000000  6.944444e-03
0.33333333  0.16666667  0.00000000  6.944444e-03
0.33333333  0.25000000  0.00000000  6.944444e-03
0.33333333  0.33333333  0.00000000  6.944444e-03
0.33333333  0.41666667  0.00000000  6.944444e-03
0.33333333  0.50000000  0.00000000  6.944444e-03
0.33333333  0.58333333  0.00000000  6.944444e-03
0.33333333  0.66666667  0.00000000  6.944444e-03
0.33333333  0.75000000  0.00000000  6.944444e-03
0.33333333  0.83333333  0.00000000  6.944444e-03
0.33333333  0.91666667  0.00000000  6.944444e-03
0.41666667  0.00000000  0.00000000  6.944444e-03
0.41666667  0.08333333  0.00000000  6.944444e-03
0.41666667  0.16666667  0.00000000  6.944444e-03
0.41666667  0.25000000  0.00000000  6.944444e-03
0.41666667  0.33333333  0.00000000  6.944444e-03
0.41666667  0.41666667  0.00000000  6.944444e-03
0.41666667  0.50000000  0.00000000  6.944444e-03
0.41666667  0.58333333  0.00000000  6.944444e-03
0.41666667  0.66666667  0.00000000  6.944444e-03
0.41666667  0.75000000  0.00000000  6.944444e-03
0.41666667  0.83333333  0.00000000  6.944444e-03
0.41666667  0.91666667  0.00000000  6.944444e-03
0.50000000  0.00000000  0.00000000  6.944444e-03
0.50000000  0.08333333  0.00000000  6.944444e-03
0.50000000  0.16666667  0.00000000  6.944444e-03
0.50000000  0.25000000  0.00000000  6.944444e-03
0.50000000  0.33333333  0.00000000  6.944444e-03
0.50000000  0.41666667  0.00000000  6.944444e-03
0.50000000  0.50000000  0.00000000  6.944444e-03
0.50000000  0.58333333  0.00000000  6.944444e-03
0.50000000  0.66666667  0.00000000  6.944444e-03
0.50000000  0.75000000  0.00000000  6.944444e-03
0.50000000  0.83333333  0.00000000  6.944444e-03
0.50000000  0.91666667  0.00000000  6.944444e-03
0.58333333  0.00000000  0.00000000  6.944444e-03
0.58333333  0.08333333  0.00000000  6.944444e-03
0.58333333  0.16666667  0.00000000  6.944444e-03
0.58333333  0.25000000  0.00000000  6.944444e-03
0.58333333  0.33333333  0.00000000  6.944444e-03
0.58333333  0.41666667  0.00000000  6.944444e-03
0.58333333  0.50000000  0.00000000  6.944444e-03
0.58333333  0.58333333  0.00000000  6.944444e-03
0.58333333  0.66666667  0.00000000  6.944444e-03
```

```
 0.58333333  0.75000000  0.00000000  6.944444e-03
 0.58333333  0.83333333  0.00000000  6.944444e-03
 0.58333333  0.91666667  0.00000000  6.944444e-03
 0.66666667  0.00000000  0.00000000  6.944444e-03
 0.66666667  0.08333333  0.00000000  6.944444e-03
 0.66666667  0.16666667  0.00000000  6.944444e-03
 0.66666667  0.25000000  0.00000000  6.944444e-03
 0.66666667  0.33333333  0.00000000  6.944444e-03
 0.66666667  0.41666667  0.00000000  6.944444e-03
 0.66666667  0.50000000  0.00000000  6.944444e-03
 0.66666667  0.58333333  0.00000000  6.944444e-03
 0.66666667  0.66666667  0.00000000  6.944444e-03
 0.66666667  0.75000000  0.00000000  6.944444e-03
 0.66666667  0.83333333  0.00000000  6.944444e-03
 0.66666667  0.91666667  0.00000000  6.944444e-03
 0.75000000  0.00000000  0.00000000  6.944444e-03
 0.75000000  0.08333333  0.00000000  6.944444e-03
 0.75000000  0.16666667  0.00000000  6.944444e-03
 0.75000000  0.25000000  0.00000000  6.944444e-03
 0.75000000  0.33333333  0.00000000  6.944444e-03
 0.75000000  0.41666667  0.00000000  6.944444e-03
 0.75000000  0.50000000  0.00000000  6.944444e-03
 0.75000000  0.58333333  0.00000000  6.944444e-03
 0.75000000  0.66666667  0.00000000  6.944444e-03
 0.75000000  0.75000000  0.00000000  6.944444e-03
 0.75000000  0.83333333  0.00000000  6.944444e-03
 0.75000000  0.91666667  0.00000000  6.944444e-03
 0.83333333  0.00000000  0.00000000  6.944444e-03
 0.83333333  0.08333333  0.00000000  6.944444e-03
 0.83333333  0.16666667  0.00000000  6.944444e-03
 0.83333333  0.25000000  0.00000000  6.944444e-03
 0.83333333  0.33333333  0.00000000  6.944444e-03
 0.83333333  0.41666667  0.00000000  6.944444e-03
 0.83333333  0.50000000  0.00000000  6.944444e-03
 0.83333333  0.58333333  0.00000000  6.944444e-03
 0.83333333  0.66666667  0.00000000  6.944444e-03
 0.83333333  0.75000000  0.00000000  6.944444e-03
 0.83333333  0.83333333  0.00000000  6.944444e-03
 0.83333333  0.91666667  0.00000000  6.944444e-03
 0.91666667  0.00000000  0.00000000  6.944444e-03
 0.91666667  0.08333333  0.00000000  6.944444e-03
 0.91666667  0.16666667  0.00000000  6.944444e-03
 0.91666667  0.25000000  0.00000000  6.944444e-03
 0.91666667  0.33333333  0.00000000  6.944444e-03
 0.91666667  0.41666667  0.00000000  6.944444e-03
 0.91666667  0.50000000  0.00000000  6.944444e-03
```

```
  0.91666667  0.58333333  0.00000000  6.944444e-03
  0.91666667  0.66666667  0.00000000  6.944444e-03
  0.91666667  0.75000000  0.00000000  6.944444e-03
  0.91666667  0.83333333  0.00000000  6.944444e-03
  0.91666667  0.91666667  0.00000000  6.944444e-03
```

Kedua *file* input di atas kemudian dieksekusi menggunakan program `pw.x` dengan perintah sebagai berikut:

```
$ mpirun -np 4 pw.x < graphene.scf.in > graphene.scf.out
$ mpirun -np 4 pw.x < graphene.nscf.in > graphene.nscf.out
```

Selanjutnya, kita siapkan *file* input untuk WANNIER90 dengan nama `graphene.win`.

```
! graphene.win

num_bands = 16
num_wann = 5

dis_win_min = -30
dis_win_max =  12
!dis_froz_min = -30
!dis_froz_max = 2.6

dis_num_iter = 5000
num_iter = 000
num_print_cycles = 100
dis_mix_ratio = 1.0
iprint = 2

begin projections
C: pz
f= 0.167, 0.333, 0.00: s
f=-0.333,-0.167, 0.00: s
f= 0.167,-0.167, 0.00: s
end projections

! EXTRACT TIGHT-BINDING PARAMETER
write_hr = .true.
write_xyz = .true.
use_ws_distance  = .true.

! PLOT BAND STRUCTURE
bands_plot = .true.
bands_num_points = 200

begin kpoint_path
```

```
G  0.000  0.000   0.000   K 0.667 -0.333  0.000
K  0.667 -0.333   0.000   M 0.500 -0.500  0.000
M  0.500 -0.500   0.000   G 0.000  0.000  0.000
end kpoint_path

begin unit_cell_cart
bohr
 4.602     0.0000000000     0.0000000000
-2.301    3.98544890821598665238     0.0000000000
 0.0000000000     0.0000000000     18.408
end unit_cell_cart

begin atoms_frac
C  0.0000000  0.0000000  0.00
C  0.3333333  0.6666667  0.00
end atoms_frac

mp_grid  = 12 12 1

begin kpoints
  ! Tulis kembali 144 k-point seperti pada file graphene.nscf.in
  0.00000000     0.00000000     0.00000000     0.00694444
  0.00000000     0.08333333     0.00000000     0.00694444
  0.00000000     0.16666667     0.00000000     0.00694444
  ... [dipotong] ...
  0.91666667     0.75000000     0.00000000     0.00694444
  0.91666667     0.83333333     0.00000000     0.00694444
  0.91666667     0.91666667     0.00000000     0.00694444
end kpoints
```

Untuk membuat agar k-point terbentang dalam wakilan *crystal*, kita bisa melakukan langkah berikut:

```
$ cd ~/wannier90.x.x.x/utility
$ ./kmesh.pl 12 12 1
```

Perintah pada baris ke-2 digunakan untuk membuat titik-titik sebanyak $12 \times 12 \times 1 = 144$, yang kemudian kita tempatkan di bagian `K_POINTS` pada *file* input untuk perhitungan `nscf` dan `Wannier`.

Selanjutnya, kita akan membahas arti pada beberapa bagian input yang kita buat. Penjelasan lengkap terkait *file* input untuk `Wannier` dapat dilihat pada *user guide* WANNIER90.[1]

- Jumlah pita dan jumlah fungsi `Wannier`.

```
num_bands = 16
num_wann = 5
```

  Variabel `num_bands` harus disesuaikan dengan jumlah pita (`nbnd`) yang dihitung oleh `nscf`, kecuali jika kita menggunakan`exclude_bands`, maka jumlah pita tidak diikutsertakan. Variable `num_wann` juga harus disesuaikan dengan banyaknya proyeksi orbital yang digunakan.
- *Outer Windows* (`dis_win_max` dan `dis_win_min`).

```
dis_win_min = -30
dis_win_max =  12
```

  Bagian maksimal dan mininal dari suatu jangkaun energi untuk mencakup seluruh pita energi untuk menggambarkan wakilan dari proyeksi orbital yang digunakan.
- Proyeksi orbital.

```
begin projections
  C: pz
  f= 0.167, 0.333, 0.00: s
  f=-0.333,-0.167, 0.00: s
  f= 0.167,-0.167, 0.00: s
end projections
```

  Bagian ini merupakan tebakan awal untuk fungsi `Wannier`, orbital sp^2 berada pada orbital π pada atom karbon dan ikatan atom C-C yang diwakilkan oleh orbital s di tengah ikatan antar karbon.
- Spesifikasi perhitungan yang diperlukan.

```
! EXTRACT TIGHT-BINDING PARAMETER
write_hr = .true.
write_xyz = .true.
use_ws_distance  = .true.

! PLOT BAND STRUCTURE
bands_plot = .true.
```

  Input di atas mewakili luaran Hamiltonian dan luaran pita energi.

Setelah perhitungan `scf` dan `nscf` selesai dilakukan, maka dilanjutkan dengan proses konversi ke perhitungan WANNIER90 dengan perintah berikut

```
$ wannier90.x -pp graphene
```

[1] http://www.wannier.org/user-guide/

Hasil dari eksekusi ini akan membangkitkan *file* `graphene.nnkp`. Tahap selanjutnya dilanjutkan dengan eksekusi *file* input `graphene.pw2wan.in`,

```
$ pw2wannier90.x < graphene.pw2wan.in > graphene.pw2wan.out
```

```
! graphene.pw2wan.in

&inputpp
    Outdir     = './tmp/'
    prefix     = 'graphene',
    seedname   = 'graphene'
    write_amn  = .true.
    write_mmn  = .true.
/
```

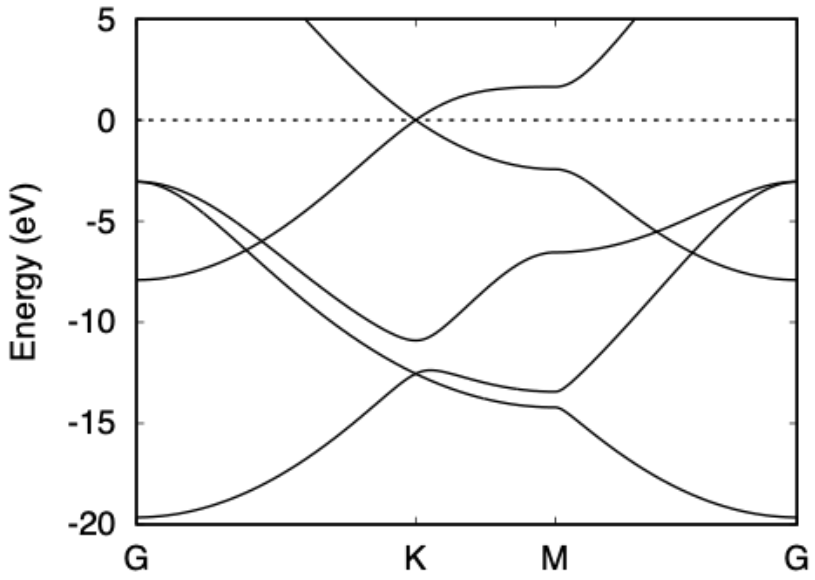
Terakhir, kita melakukan eksekusi fungsi gelombang Wannier terlokalisir maksimal,

```
$ wannier90.x graphene
```

Hasil dari eksekusi tersebut, kita akan disuguhkan informasi perihal pita energi yang diwakilkan oleh luaran `graphene_band.gnu` dan Hamiltonian model i dengan fungsi `Wannier` diwakilkan oleh luaran `graphene_hr.dat`. Kita bisa melakukan *plotting* pita energi dari luaran `graphene_band.gnu` dengan cara sebagai berikut:

```
gnuplot > load 'graphene_band.gnu'
```

Sehingga kita peroleh gambar seperti di bawah ini:



Apabila kita inspeksi luaran `graphene_hr.dat`, kita bisa mendapati paramater *tight-binding*, seperti integral transfer untuk tetangga terdekat (t) dan integral transfer untuk tetangga terdekat berikutnya (t'), yang masing-masing bernilai -3.00 dan -0.23 eV.

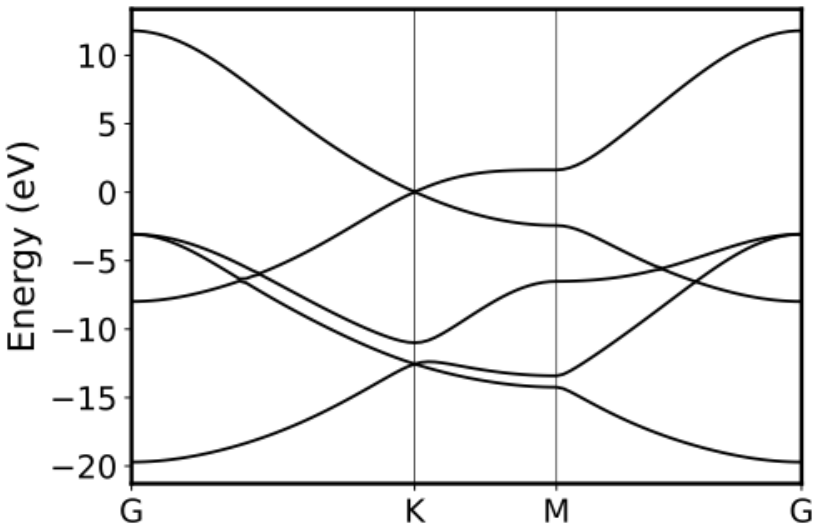
```
  0   0   0   5   1    0.000000   -0.000000
  0   0   0   1   2   -3.001299    0.000000 <--
  0   0   0   2   2   -0.154584   -0.000000
  0   0   0   3   2    0.000000   -0.000000
  ...
  0   0   0   5   5  -11.355928    0.000000
  0   1   0   1   1    0.231744   -0.000000 <--
  ...
```

Kemudian, kita bisa membaca luaran `graphene_hr.dat` dengan bantuan PYTHTB[2] untuk menyelesaikan diagonalisasi matriks dengan membaca paramater-paramater *tight-binding* dari perhitungan WANNIER90. Di bawah ini merupakan *script* untuk menselesaikan diagonalisasi matriks untuk mendapatkan pita energi:

```
from pythtb import * # import TB model class
import matplotlib.pyplot as plt
# load data graphene_hr.dat
graphene = w90('./', 'graphene')
# Fermi level
EFermi = -0.4658
# Contruct TB model
my_model = graphene.model(zero_energy = EFermi, min_hopping_norm
    =0.01)
# solve model on a path and plot it
path=[ [0.000, 0.000, 0.000],[0.667,-0.333, 0.000],[0.500, -0.500,
    0.000], [0.000, 0.000, 0.000] ]
# labels of the nodes
k_label=(r'G', r'K', r'M', r'G')
# call function k_path to construct the actual path
(k_vec,k_dist,k_node)=my_model.k_path(path,101)
evals=my_model.solve_all(k_vec)
fig, ax = plt.subplots(figsize=(6,4))
for i in range(evals.shape[0]):
    ax.plot(k_dist,evals[i],"k-")
for n in range(len(k_node)):
    ax.axvline(x=k_node[n],linewidth=0.5, color='k')
ax.set_ylabel("Energy (eV)")
ax.set_xlim(k_dist[0],k_dist[-1])
ax.set_xticks(k_node)
ax.set_xticklabels(k_label)
fig.tight_layout()
fig.savefig('./TBmodel.svg')
```

[2]https://www.physics.rutgers.edu/pythtb/

Dari *script* PYTHON di atas kita akan mendapatkan gambar pita energi dari parameter *tight-binding* untuk material graphene.



5 | Gnuplot dan Visualisasi Data

oleh: Ahmad R. T. Nugraha

Sebagai pelajar ataupun peneliti, kita dihadapkan pada beragam pilihan perangkat lunak pembuat grafik, kurva, histogram, serta bentuk-bentuk gambar lainnya yang diperlukan dalam menyajikan data hasil penelitian ataupun untuk menggambar suatu fungsi matematis tertentu. Perangkat visualisasi yang kita gunakan sering kali ditentukan berdasarkan kemudahan mengoperasikannya, hasil keluaran (output) gambar yang cocok dengan format makalah atau format suatu jurnal tertentu, serta harganya, apakah mahal, murah, atau justru gratis.

Beberapa perangkat visualisasi yang cukup populer karena kemampuan dan kemudahan penggunaannya dalam pembuatan bermacam-macam grafik di antaranya adalah Microsoft Excel, OriginPro, dan Igor Pro. Sayangnya, semua perangkat ini bersifat komersial (berbayar) dengan harga yang cukup mahal untuk kantong pelajar dan peneliti di Indonesia.

Memang, ada beberapa versi "tiruan" yang gratis dari perangkat yang disebutkan di atas, seperti LibreOffice Calc (mirip Microsoft Excel) dan QtiPlot/SciDAVis (mirip OriginPro dan Igor Pro). Namun, bisa jadi kita tidak puas jika sekadar menggunakan "software KW1" meskipun sebenarnya perangkat-perangkat tersebut punya kemampuan yang cukup baik.

Ada satu kesamaan karakteristik dari Excel, Origin, Igor, serta varian perangkat lunak yang serupa. Perangkat-perangkat ini pada dasarnya menggunakan antarmuka grafis untuk menampilkan gambar yang sedang kita buat. Ketika menggunakan perangkat-perangkat tersebut, kita bisa langsung melihat hasil akhir gambar yang kira-kira akan terbentuk dari proses pengolahan data yang kita lakukan.

Pada tutorial ini, kita tidak akan membahas perangkat visualisasi yang menggunakan antarmuka grafis, tetapi kita coba ulas jenis perangkat visualisasi yang menggunakan antarmuka teks, yaitu Gnuplot. Dengan menggunakan antarmuka teks, Gnuplot memiliki keunggulan dalam hal kecepatan pengolahan gambar dibandingkan perangkat sejenis yang menggunakan antarmuka grafis.

Dalam Gnuplot, kita cukup mengetik beberapa baris kode, yang selanjutnya diterjemahkan oleh Gnuplot menjadi suatu bentuk grafik. Selain itu, Gnuplot juga

merupakan salah satu perangkat visualisasi gratis (tak berbayar) yang memiliki kemampuan terlengkap untuk menggambar hampir segala macam grafik, seperti plot 2D, 3D, plot permukaan, peta, beragam fungsi matematis, fitting kurva, hingga animasi.

GNUPLOT dapat memberikan format keluaran (output) gambar yang berbeda-beda seperti *portable document format* (PDF) yang merupakan tipe gambar vektor serta *portable network graphics* (PNG) yang merupakan tipe gambar *bitmap*. Format PDF paling sering dipakai dalam pembuatan dokumen dengan LaTeX, yang biasanya menjadi standar penulisan makalah di banyak jurnal. Format PNG biasanya digunakan pada pengolah dokumen yang bersifat WYSIWIG (*what you see is what you get*) seperti Microsoft Word maupun digunakan pada pembuatan situs web.

Untuk keperluan penyajian data penelitian sehari-hari, kita hanya perlu mengetahui beberapa kode dasar GNUPLOT. Oleh karenanya, fokus pembahasan pada tutorial ini dibatasi pada bagaimana menghasilkan grafik dari suatu data 2D (yang memiliki sumbu-x, sumbu-y, serta *error bar*) dan bagaimana menggambar fungsi matematis sederhana. Kemampuan GNUPLOT lebih lanjut dapat dieksplorasi sendiri dari buku manualnya.

5.1 Instalasi GNUPLOT

Langkah pertama yang perlu dilakukan sebelum menggunakan GNUPLOT tentunya adalah melakukan instalasi perangkat ini di komputer kita. Bagi pengguna Windows, versi terbaru GNUPLOT dapat diunduh dari laman:

https://sourceforge.net/projects/gnuplot/files/gnuplot/

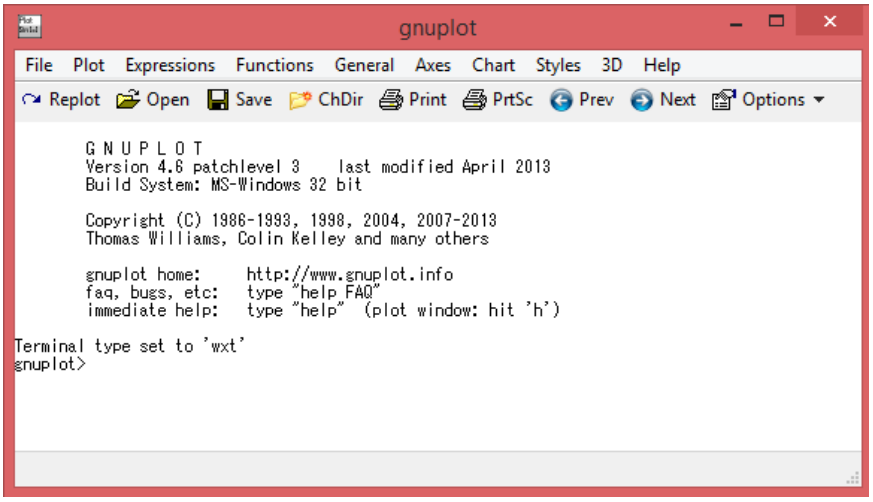
Pastikan kita mengunduh berkas instalasi dengan ekstensi *.mingw.exe. Misalnya, pada waktu tutorial ini ditulis, berkas yang bisa diunduh pengguna Windows adalah `gp542-win64-mingw.exe`.

Bagi pengguna Linux, bergantung pada distribusi Linux yang digunakan, kita bisa instalasi GNUPLOT langsung dari terminal, misalnya dengan perintah berikut ini untuk distribusi berbasis Ubuntu/Debian:

```
sudo apt install gnuplot
```

Satu lagi perangkat tambahan yang dapat membantu penanganan beragam format gambar keluaran GNUPLOT adalah Inkscape [1]. Inkscape pada dasarnya merupakan pengolah format *scalable vector graphics* (SVG) sehingga bisa digunakan untuk memodifikasi gambar PDF yang telah dihasilkan oleh GNUPLOT. Inkscape pun mampu mengubah satu format gambar (vektor ataupun *bitmap*) dengan menyimpannya menjadi format gambar yang lain.

[1] http://www.inkscape.org/



Gambar 5.1: Tampilan GNUPLOT untuk Windows.

5.2 Penggunaan Dasar GNUPLOT

Dalam sebuah sesi GNUPLOT, biasanya kita perlu persiapkan dua perangkat sekaligus. Selain GNUPLOT, kita akan membutuhkan editor teks semacam Notepad yang digunakan untuk mengetikkan kode-kode GNUPLOT.

Untuk contoh pertama, mari kita buat grafik dari sebuah data 2D. Format data 2D yang dapat diterima Gnuplot adalah format teks biasa, seperti *.csv atau *.dat. Dalam tutorial ini kita akan selalu menyimpan data dalam format *.dat.

Kita dapat membuat sembarang data 2D bertipe *.dat, baik itu dari hasil penelitian atau sekadar data sederhana yang mengandung sumbu-x dan sumbu-y seperti ini:

```
# data1.dat
# X Y
    1 5
    2 3
    3 2
    4 4
```

Setiap baris teks yang diawali oleh tanda # berfungsi sebagai "komentar" dan tidak akan diolah oleh GNUPLOT. Perhatikan pada contoh data ini kolom X dan Y dapat dipisahkan oleh suatu "tab", "spasi", ataupun "koma". GNUPLOT akan menganggap setiap separasi tersebut sebagai tanda untuk kolom-kolom yang berbeda.

Jika kita sudah punya suatu berkas data 2D, kita dapat memulai menggambar grafiknya menggunakan GNUPLOT. Misalkan data tersebut kita simpan dengan nama 'data1.dat' pada folder 'E:\gnuplot' (Windows) atau pada folder '/home/user/gnuplot' (Linux). Kita dapat

buka perangkat GNUPLOT di Windows maupun Linux dan mulai memberikan instruksi pada GNUPLOT.

Untuk menuju ke folder tempat data yang kita inginkan, gunakan perintah cd (change directory) pada terminal GNUPLOT:

```
gnuplot> cd 'E:\gnuplot'
```

atau

```
gnuplot> cd '/home/user/gnuplot'
```

sesuai sistem operasi yang digunakan. Tanda petik pada terminal GNUPLOT di atas boleh menggunakan **petik tunggal** ataupun **petik ganda**.

Kita kemudian bayangkan data 2D yang sudah kita definisikan akan digambar sebagai titik-titik data yang dihubungkan oleh suatu garis. Untuk menghasilkan gambar tersebut, kita perlu "memberi tahu" GNUPLOT terkait definisi *style* yang ingin digunakan. Bentuk kode dasarnya adalah:

```
set style line <index> lt <linetype> lc <linecolor> \ lw <linewidth>
  pt <pointtype> ps <pointsize>
```

Sebagai contoh, pada terminal GNUPLOT dapat kita masukkan:

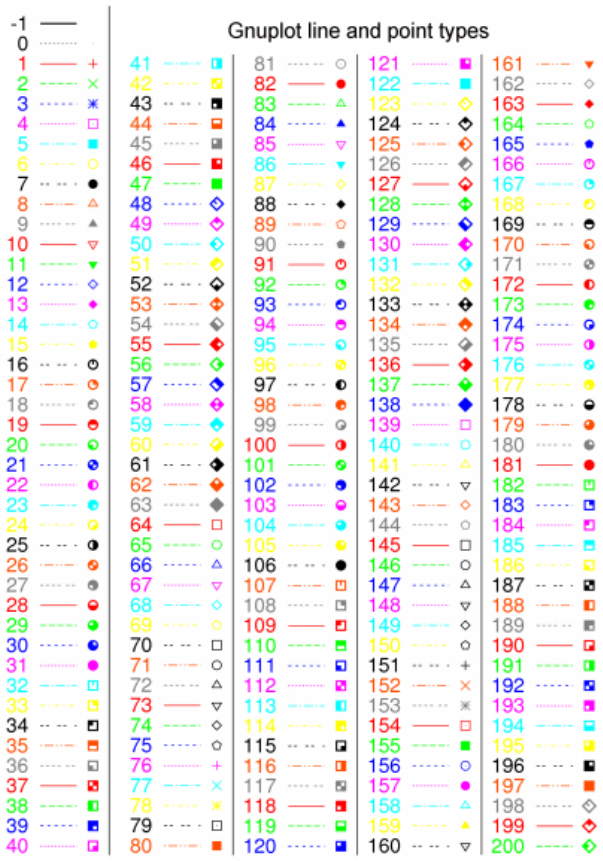
```
gnuplot> set style line 1 lt 1 lc 3 lw 2 pt 7 ps 2
```

Dalam kode ini, `<index>` = 1 merupakan penanda bahwa style yang didefinisikan pada contoh ini diberi nomor 1 untuk dirujuk pada perintah-perintah GNUPLOT berikutnya. Pemberian nilai `<index>` kelak sangat berguna jika kita memiliki beragam data yang ingin kita plot dalam satu gambar yang sama namun dengan *style* yang berbeda.

Setiap spesifikasi `<linetype>`, `<linecolor>`, dan seterusnya, yang dinyatakan setelah `<index>` akan selalu terkait dengan nomor indeks tersebut. Pada contoh tadi, spesifikasi nilai `<linetype>` = 1 merupakan definisi garis lurus, kemudian `<linecolor>` = 3 adalah warna biru untuk garis tersebut, `<linewidth>` = 2 adalah ketebalan garis sebesar 2 titik, `<pointtype>` = 7 mendefinisikan titik data yang digambarkan berupa bulatan, dan `<pointsize>` = 2 merupakan ketebalan bulatan sebesar 2 titik.

Kita dapat memberikan nilai yang lain untuk setiap definisi `<linetype>`, `<linecolor>` dan `<pointtype>` sesuai dengan tampilan yang kita inginkan. Gambar 5.2 menunjukkan rangkuman beberapa bentuk garis, titik data, dan warna yang dapat dipilih pada GNUPLOT.

Khusus untuk pilihan warna, kita bahkan dapat mendefinisikan warna sendiri yang lebih banyak pilihannya daripada yang tersedia pada Gambar 5.2. Caranya adalah dengan menggunakan kode HTML untuk warna dalam format RGB. Pada contoh yang telah kita berikan, nilai `<linecolor>` = 3 dapat diganti dengan `<linecolor>` = `rgb '#0060ad'`:



Gambar 5.2: Angka yang dapat dipilih untuk bentuk garis, titik data, dan warna tertentu pada GNUPLOT.

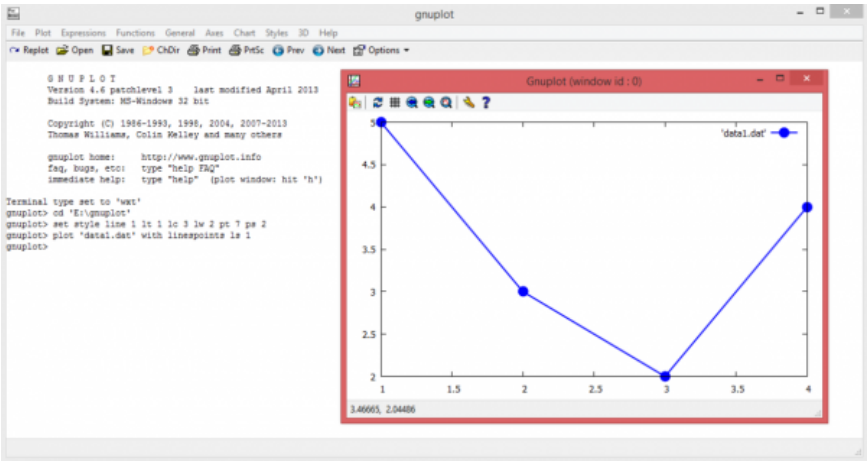
```
gnuplot> set style line 1 lt 1 lc rgb '#0060ad' lw 2 pt 7 ps 2
```

Setelah mendefinisikan penampilan data dalam bentuk garis dan bentuk titik tertentu, kita dapat langsung menggambar data 2D dengan perintah *plot*:

```
gnuplot> plot 'data1.dat' with linespoints ls 1
```

Perhatikan bahwa `<linestyle>` = 1 merupakan jenis style yang sudah didefinisikan sebelumnya. Spesifikasi '`with linespoints`' merupakan perintah kepada GNUPLOT agar menggambar data dengan menghubungkannya menggunakan garis sekaligus setiap koordinat diberikan titik data dengan bentuk yang sudah ditentukan. Jika kita menginginkan agar data digambar dengan garis saja, kita bisa gunakan spesifikasi '`with lines`'. Di sisi lain, untuk data yang digambar dengan titik-titik saja, kita gunakan '`with points`'.

Dengan mengombinasikan semua informasi dasar ini dalam sebuah sesi GNUPLOT, kita dapat memperoleh tampilan seperti pada Gambar 5.3.



Gambar 5.3: Kombinasi penggunaan beberapa kode dasar GNUPLOT beserta hasilnya.

Untuk menyimpan hasil plot dalam sebuah berkas bertipe PDF atau PNG, kita perlu mendefinisikan format keluaran gambar di dalam GNUPLOT, yaitu definisi tipe gambar dan nama gambar keluaran. Definisi keluaran ini sebaiknya dimasukkan di awal sebelum memulai definisi *style* dari gambar yang ingin dibuat. Berdasarkan tipe gambar yang ingin dihasilkan, kode yang dibutuhkan pun sedikit berbeda, seperti berikut ini:

PNG

```
set terminal pngcairo size 400,300 enhanced font 'Helvetica,12'
set output 'plotdata1.png'
```

PDF

```
set terminal pdfcairo size 8cm,6cm font 'Helvetica,12'
set output 'plotdata1.pdf'
```

Perhatikan pada definisi keluaran PNG, nilai 400,300 merupakan ukuran gambar yang ingin ditampilkan dalam satuan pixels, sedangkan pada tipe gambar PDF kita bisa memberikan spesifikasi dalam satuan cm. Penggunaan satuan lainnya dapat dicek lebih lanjut pada manual GNUPLOT. Dalam mendefinisikan format keluaran ini kita juga dapat memberikan jenis huruf tertentu beserta ukuran hurufnya. Kita bisa coba-coba ukuran yang kita kehendaki dan sesuaikan dengan selera artistik masing-masing sesuai standar ilmiah.

Sekarang, alih-alih mengetikkan setiap kode GNUPLOT satu per satu via terminal GNUPLOT seperti yang dilakukan sebelumnya pada Gambar 3, kita bisa mempersingkat proses menggambar grafik dengan memasukkan semua kode GNUPLOT yang dibutuhkan pada sebuah teks yang disimpan dalam format *.plt, *.gp, atau *.gpl, kemudian membukanya belakangan dari terminal GNUPLOT. Ada tiga langkah mendasar yang perlu dilakukan:

1. Buka editor teks seperti Notepad untuk membuat sebuah berkas baru. Untuk contoh

grafik yang telah diberikan, kita dapat masukkan kode berikut ke dalam editor teks dan menyimpannya sebagai ‘plotdata1.plt’:

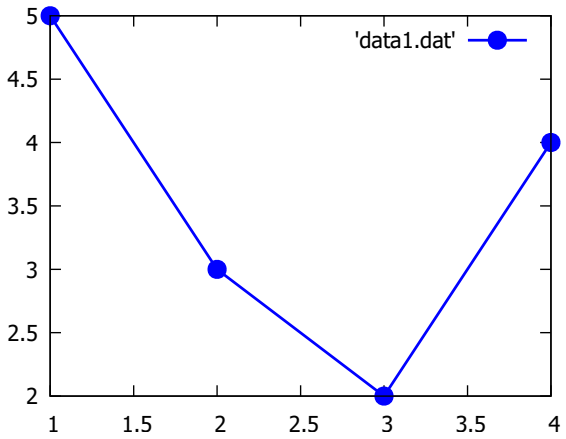
```
# plotdata1.plt
reset
set terminal pdfcairo size 8cm,6cm font ’Helvetica,12’
set output ’plotdata1.pdf’
set style line 1 lt 1 lc 3 lw 2 pt 7 ps 1
plot ’data1.dat’ with linespoints ls 1
unset output
```

Berkas ‘plotdata1.plt’ ini disimpan dalam folder yang sama dengan ‘data1.dat’. Keluarannya juga (pada contoh ini adalah tipe gambar PDF: ‘plotdata1.pdf’) akan diperoleh dalam folder yang sama.

2. Dari terminal GNUPLOT, masukkan perintah load untuk kode GNUPLOT yang telah disimpan. Asumsikan kita menyimpannya dalam folder ‘E:\gnuplot’:

```
gnuplot> cd ’E:\gnuplot’
gnuplot> load ’plotdata1.plt’
```

3. Jika tidak ada pesan *error*, gambar yang diinginkan sudah dapat dilihat seperti pada Gambar 5.4.



Gambar 5.4: Grafik ‘plotdata1.pdf’ keluaran GNUPLOT.

5.3 Impor, Modifikasi, dan Ekspor Gambar dengan Inkscape

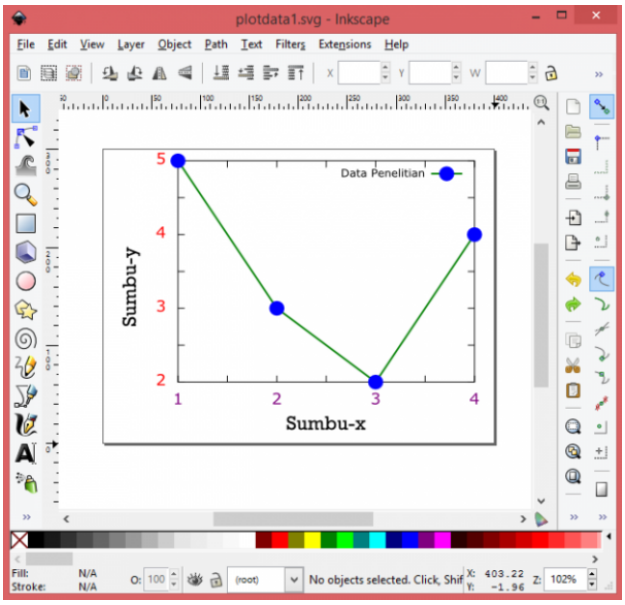
Penggunaan Inkscape sangat bermanfaat untuk memodifikasi lebih lanjut gambar yang telah dihasilkan GNUPLOT jika kita masih tidak puas dengan kualitas gambar keluaran GNUPLOT.

Modifikasi yang dimaksud di sini jangan sampai mengubah data hasil penelitian, yakni sekadar mengubah warna, ukuran huruf, dan hal-hal terkait penampilan gambar secara artistik.

Kita dapat mengimpor terlebih dahulu gambar yang telah dihasilkan oleh GNUPLOT tadi melalui menu **File** → **Import** di dalam Inkscape. Selanjutnya, kita simpan dengan ekstensi berkas *.svg, misalnya beri nama berkasnya `'plotdata1.svg'` via **File** → **Save as**. Kita dapat memodifikasi atau memperbaiki gambar sesuka kita sambil selalu memperbarui dan menyimpannya.

Setelah selesai urusan memperbaiki gambar, Inkscape dapat menangani ekspor gambar ke dalam berbagai format. Untuk tipe gambar bitmap, kita bisa pilih **File** → **Export Bitmap**. Kemudian, tulis nama dan tipe keluaran yang diinginkan (PNG, JPG, BMP, dll). Untuk ekspor ke tipe PDF, kita perlu memilih **File** → **Save a Copy**. Dari panel yang muncul kita pilih **"Save as Type"** sesuai format gambar yang diinginkan.

Sebagai contoh, seperti ditunjukkan pada Gambar 5.5, kita dapat mengubah penampilan `'plotdata1.pdf'` (yang sudah diganti namanya menjadi `'plotdata1.svg'`(pada Inkscape dengan menambah atau menghapus teks, mengganti warna, memperkecil ataupun memperbesar elemen gambar. Setelah itu, kita dapat menyimpan gambar ke dalam format keluaran yang kita inginkan.



Gambar 5.5: Permak grafik `'plotdata1.pdf'` dengan menggunakan Inkscape.

Meski sedikit merepotkan, tanpa Inkscape, kita bisa juga menyuruh GNUPLOT menghasilkan keluaran PDF dan PNG sekaligus dengan tambahan beberapa baris kode:

```
# ...
# lanjutan plotdata1.plt
set terminal pngcairo size 400,300 enhanced font 'Helvetica,12'
```

```
set output 'plotdata1.png'
replot
unset output
```

5.4 Variasi Bentuk Data 2D

Pada contoh pertama yang telah dijelaskan, kita hanya menggambar grafik dari sebuah data 2D yang mengandung satu himpunan nilai x dan satu himpunan nilai y. Selain bentuk data semacam itu, GNUPLOT dapat mengenali secara otomatis jika antara suatu himpunan data tertentu dan himpunan data lainnya dipisahkan oleh sebuah baris kosong.

Untuk contoh kasus ini, kita modifikasi 'data1.dat' dengan menekan tombol 'Enter' setelah baris data kedua sehingga baris tersebut terpisah dari baris data ketiga.

```
# data2.dat
# X Y
  1 5
  2 3

  3 2
  4 4
```

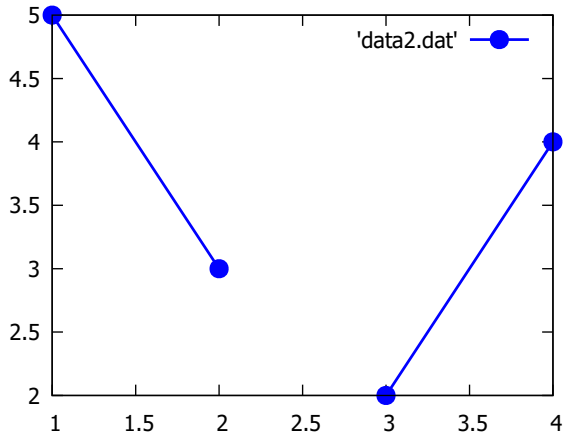
Kita simpan data baru ini dengan nama 'data2.dat'. Selanjutnya, kita ubah juga 'plotdata1.plt' dengan mengganti ekspresi 'data1.dat' menjadi 'data2.dat', 'plotdata1.pdf' menjadi 'plotdata2.pdf', dan teks kode ini disimpan dengan nama 'plotdata2.pdf'.

```
# plotdata2.plt
reset
set terminal pdfcairo size 8cm,6cm font 'Helvetica,12'
set output 'plotdata2.pdf'
set style line 1 lt 1 lc 3 lw 2 pt 7 ps 1
plot 'data2.dat' with linespoints ls 1
unset output
```

Eksekusi 'plotdata2.plt' dari terminal GNUPLOT:

```
gnuplot> cd 'E:\gnuplot'
gnuplot> load 'plotdata2.plt'
```

Hasilnya seperti seperti ditunjukkan pada Gambar 5.6. Perhatikan bahwa sekarang ada titik-titik data yang tidak terhubungkan oleh garis karena titik-titik data tersebut dianggap sebagai bagian dari blok data yang berbeda.



Gambar 5.6: Keluaran dari ‘plotdata2.plt’, yaitu ‘plotdata2.pdf’.

Jika kita ingin memberi warna yang berbeda untuk dua blok data ini, kita dapat mengubah bentuk ‘data2.dat’ lebih jauh lagi. Kali ini antara blok data pertama dan blok data kedua di dalam ‘data2.dat’ kita pisahkan dengan dua baris dan kita beri nomor indeks (index) tertentu. Nama berkas ini pun kita ganti menjadi ‘data3.dat’.

```
# data3.dat
# Blok pertama (index 0)
  1 5
  2 3

# Blok kedua (index 1)
  3 2
  4 4
```

Teks ‘plotdata2.plt’ kemudian dimodifikasi menjadi ‘plotdata3.plt’ dengan menambahkan *style* untuk data yang berbeda. Kita dapat merujuk pada Gambar 5.2 untuk memilih warna, tipe garis, dan tipe titik data yang dapat dipilih.

Pada contoh ini, kita ingin memberi bentuk kotak (<pointtype> = 5) untuk koordinat data pada blok kedua, serta warna merah (<linecolor> = 1) pada garis dan koordinat data. Baris kode yang perlu ditambahkan adalah:

```
set style line 2 lt 1 lc 1 lw 1 pt 5 ps 1
```

Perintah *plot* pun perlu dimodifikasi sesuai blok data yang hendak digambar:

```
plot 'data3.dat' index 0 with linespoints ls 1,\
     'data3.dat' index 1 with linespoints ls 2
```

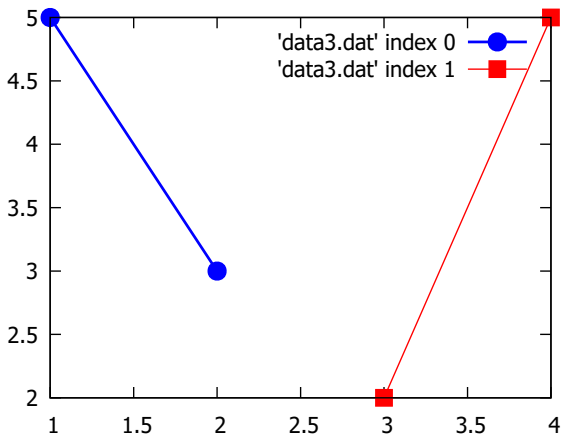
Perhatikan bagaimana posisi index diatur sesuai dengan definisi style yang diinginkan. Implementasi lengkapnya dapat dilihat pada ‘plotdata3.plt’.

```
# plotdata3.plt
reset
set terminal pdfcairo size 8cm,6cm font 'Helvetica,12'
set output 'plotdata3.pdf
set style line 1 lt 1 lc 3 lw 2 pt 7 ps 1
set style line 2 lt 1 lc 1 lw 1 pt 5 ps 1
plot 'data3.dat' index 0 with linespoints ls 1, \
     'data3.dat' index 1 with linespoints ls 2
unset output
```

Eksekusi 'plotdata3.plt' ini pada terminal GNUPLOT:

```
gnuplot> cd 'E:\gnuplot'
gnuplot> load 'plotdata3.plt'
```

Hasilnya dapat dilihat pada Gambar 5.7.



Gambar 5.7: Keluaran dari 'plotdata3.plt', yaitu 'plotdata3.pdf'.

5.5 Penataan Sumbu, Legenda, dan Judul dari suatu Grafik

Sampai sejauh ini, kita masih belum memanfaatkan kemampuan GNUPLOT dalam menspesifikasi pengaturan sumbu, legenda, ataupun judul grafik. Pada dasarnya, label sumbu dan judul grafik tidak akan diberikan jika kita tidak spesifikasi, sedangkan label legenda langsung diberikan sesuai dengan nama data yang ada. Kita dapat mengatur elemen-elemen tersebut sesuai keinginan dengan beberapa kode tambahan.

Kita bisa ambil contoh dari 'plotdata3.plt'. Misalkan sekarang kita ingin data yang divisualisasikan diberi label sumbu-x sebagai 'Waktu (bulan)' dan sumbu-y sebagai

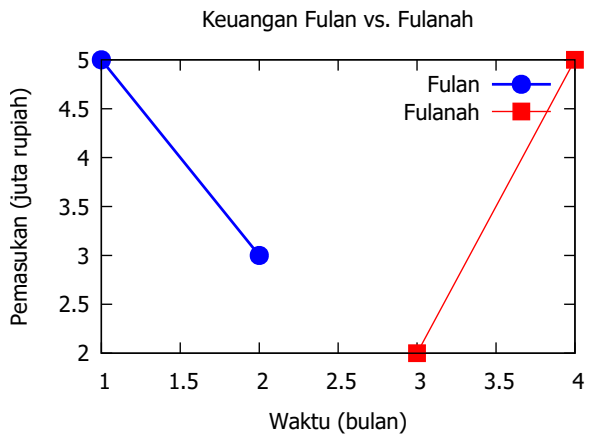
'Pemasukan (juta rupiah)'. Kemudian, data dengan `index 0` diberi label 'Fulan', sedangkan data dengan `index 1` diberi label 'Fulanah'. Judul grafiknya sendiri adalah 'Keuangan Fulan versus Fulanah' Berdasarkan kebutuhan tersebut, kode-kode yang perlu dieksekusi ditunjukkan pada teks '`plotdata3-rev.plt`'.

```
# plotdata3.plt
reset
set terminal pdfcairo size 8cm,6cm font 'Helvetica,12'
set output 'plotdata3-rev.pdf'
set style line 1 lt 1 lc 3 lw 2 pt 7 ps 1
set style line 2 lt 1 lc 1 lw 1 pt 5 ps 1
set xlabel 'Waktu (bulan)'
set ylabel 'Pemasukan (juta rupiah)'
set title  'Keuangan Fulan vs. Fulanah'
plot 'data3.dat' index 0 title 'Fulan'   with linespoints ls 1, \
     'data3.dat' index 1 title 'Fulanah' with linespoints ls 2
unset output
```

Seperti biasa, kode ini dieksekusi pada terminal GNUPLOT:

```
gnuplot> cd 'E:\gnuplot'
gnuplot> load 'plotdata3-rev.plt'
```

Hasilnya dapat dilihat pada Gambar 5.8.



Gambar 5.8: Keluaran dari '`plotdata3-rev.plt`'. Label sumbu, judul, dan legenda telah ditambahkan pada grafik langsung melalui perintah GNUPLOT.

Andaikata kita tidak ingin memberi label legenda (yang secara default muncul pada grafik), kita dapat meminta GNUPLOT untuk menghilangkan label tersebut dengan menyelipkan perintah `unset key` sebelum perintah `plot`. Di samping itu, jika kita masih tidak puas dengan hasil keluaran gambar dari GNUPLOT, kita selalu dapat memperbaikinya dengan Inkscape.

5.6 Visualisasi Data dengan *Error Bar*, *Fitting*, dan Fungsi Matematis

Untuk contoh terakhir yang diberikan dalam tutorial ini, kita akan melihat bagaimana GNUPLOT dapat memvisualisasikan data yang mengandung *error bar*, sekaligus melakukan *fitting* dan menggambarkan fungsi matematis tertentu pada sebuah grafik. Kita ambil contoh data peluruhan Ag-108 [2] dari dokumentasi GNUPLOT:

```
#
# $Id: silver.dat,v 1.1 1998/04/15 19:16:42 lhecking Exp $
#
10.000000   280.000000   16.733201
20.000000   191.000000   13.820275
# [dst... data lengkapnya dipotong]
590.000000    9.000000    3.000000
600.000000    5.000000    2.236068
```

Struktur data ‘silver.dat’ ini tersusun atas data waktu pada kolom pertama, laju peluruhan pada kolom kedua, serta ketidakpastian laju peluruhan (*error bar*) pada kolom ketiga. Kita inginkan keluaran gambar mengandung 3 macam kurva, yaitu data eksperimen, interpolasi polinom dengan algoritma tertentu, dan sebuah fungsi matematis yang menggambarkan tren data eksperimen secara kasar.

```
# plotsilver.plt
reset
set terminal pdfcairo size 8cm,6cm font 'Helvetica,12'
set output 'plotsilver.pdf'
set style line 1 lt 1 lc rgb 'blue'  lw 1.5 pt 7 ps 0.5
set style line 2 lt 1 lc rgb 'red'   lw 1.5
set style line 3 lt 5 lc rgb 'green' lw 1.5
set xlabel 'Waktu (detik)'
set ylabel 'Laju Peluruhan'
set title  'Data Laju Peluruhan Ag-108'
# rentang sumbu
set xrange [0:250]
set yrange [0:300]
set mxtics 5
set mytics 5
# rentang fitting:
set samples 300,300
# fungsi matematis:
T(x) = 12.6 + 313*exp(-x/42.7)
# plot data, fitting, dan fungsi:
plot 'silver.dat' title 'Data Eksperimen' with errorb    ls 1, \
```

[2] https://github.com/gnuplot/gnuplot/blob/master/demo/silver.dat

```
     'silver.dat' title 'Bezier Fitting' smooth sbezier  ls 2, \
     T(x)         title 'Fungsi Tebakan'  with lines     ls 3
unset output
```

Pada kode ini, kita telah menggunakan perintah baru seperti `set samples`, pendefinisian fungsi matematis, dan opsi penggambaran *error bar* pada perintah `plot`. Perintah `set samples` menentukan jumlah data yang akan diambil dalam proses *fitting*. Di sini *fitting* dilakukan dalam bentuk interpolasi polinom dengan algoritma Bezier. Baris perintah gambar yang terkait adalah:

```
plot 'silver.dat' smooth sbezier
```

GNUPLOT akan secara otomatis menginterpolasi data eksperimen dengan algoritma tersebut. Ada lebih banyak detail seputar *fitting* pada GNUPLOT yang dapat dibaca pada manual. Tutorial ini sekadar sebagai perangsang untuk sedikit melihat kemampuan GNUPLOT dalam proses *fitting*.

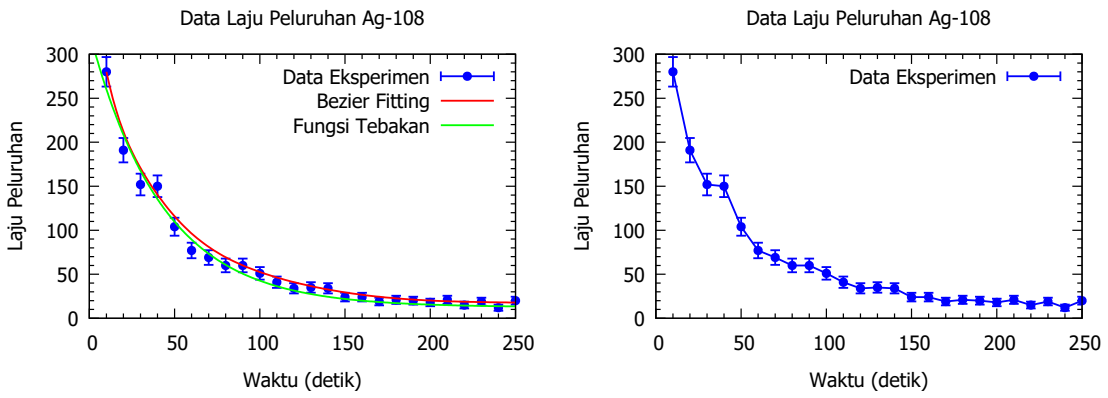
Selanjutnya, dengan GNUPLOT kita bisa mendefinisikan fungsi matematis sesuai yang kita kehendaki. Untuk gambar 2D, fungsi matematis dinotasikan sebagai $f(x)$, yaitu fungsi 1 variabel. Rentang variabel x yang digunakan disesuaikan dengan perintah `set xrange` yang telah dituliskan terlebih dahulu. Operasi-operasi matematis yang dapat digunakan pada GNUPLOT di antaranya mencakup operasi aritmetika, operasi trigonometri, dan eksponensial. Beberapa operasi matematis lebih lanjut juga dapat dirujuk pada manual.

Setelah mengeksekusi '`plotsilver.plt`' ini pada terminal GNUPLOT, kita bisa peroleh hasil seperti pada Gambar 9 di bagian kiri. Panel kanan pada Gambar 5.9 menunjukkan variasi keluaran '`plotsilver.plt`' jika kita ganti perintah plot di akhir '`plotsilver.plt`' dengan baris kode berikut ini:

```
plot 'silver.dat' title 'Data Eksperimen' with errorb    ls 1, \
     'silver.dat' using 1:2 notitle       with  lines    ls 1
```

Pada baris pertama kode di atas, kita menjaga perintah `plot` seperti semula karena kita ingin menggambar titik-titik data dengan *error bar*. Opsi `notitle` digunakan karena kita tidak ingin legenda data ini muncul lagi pada gambar. Sementara itu, opsi `using 1:2` pada baris kedua mengindikasikan bahwa untuk bagian plot tersebut kita hanya mengambil kolom 1 dan kolom 2 dari data '`silver.dat`' dan kemudian digambar garisnya dengan opsi `with lines`. Itulah sebabnya kita bisa lihat antar titik-titik data yang mengandung *error bar* pada bagian kanan Gambar 5.9 dapat dihubungkan oleh sebuah garis.

Penggunaan opsi `using` lebih lanjut pada perintah `plot` sangat bermanfaat jika kita memiliki data mentah penelitian yang mengandung banyak kolom, padahal kita hanya ingin memvisualisasikan kolom-kolom tertentu saja dari data tersebut. Alih-alih mengubah data mentah dan menyalinnya pada berkas data baru, kita cukup memilih kolom tertentu yang ingin kita presentasikan dari data mentah yang ada.



Gambar 5.9: Keluaran '`plotsilver.plt`' (panel kiri) dan variasinya (panel kanan).

Demikianlah tutorial singkat GNUPLOT ini beserta beberapa perangkat pendukungnya untuk keperluan visualisasi data penelitian. Kita dapat mempelajari bermacam-macam grafik yang dapat dibuat oleh GNUPLOT seperti yang telah disebutkan di awal dengan banyak melakukan coba-coba sendiri untuk setiap perintah/kode GNUPLOT yang tersedia pada manual.

Dengan banyak berlatih, kita dapat mengingat kode-kode `Gnuplot` secara otomatis yang kemudian mempercepat proses penyelesaian suatu gambar dan visualisasi data. Terlebih lagi, `Gnuplot` versi terbaru sudah mendukung beragam sintaks terstruktur layaknya sebuah bahasa pemrograman. Semakin sering kita berinteraksi dengan kode-kode ini, semakin mudah grafik-grafik yang sulit sekalipun untuk kita gambar dalam waktu singkat.

Cita rasa seni yang baik dapat terasah dengan melihat beragam contoh yang ada pada situs-situs demo `Gnuplot`. Tentunya kita ingin agar gambar hasil penelitian yang kita buat tidak hanya menampilkan data semata-mata, tetapi juga enak dipandang dan memiliki kesan visual yang mendalam.

Referensi lebih lanjut:

- Demo GNUPLOT: http://gnuplot.sourceforge.net/demo/
- Manual GNUPLOT: http://www.gnuplot.info/documentation.html
- Tip dan trik GNUPLOT: http://www.gnuplotting.org/
- GNUPLOT Not So FAQ: http://lowrank.net/gnuplot/index-e.html
- Arsip GNUPLOT pada Wikipedia:
  http://commons.wikimedia.org/wiki/Category:Gnuplot$_$diagrams

6 | *Symbolic* Python

OLEH: M. SHOUFIE UKHTARY

Selain digunakan untuk perhitungan numerik, PYTHON dapat digunakan juga untuk perhitungan simbolik (*analytical calculation*). Perhitungan simbolik PYTHON dapat dilakukan dengan menggunakan *library* `SymPy`. *Library* ini memiliki fitur seperti program Mathematica, tetapi dapat digunakan secara gratis. `SymPy` sangat berguna saat kita memerlukan ekspresi analitik dari suatu perhitungan, seperti turunan dan integral dari suatu fungsi atau menyelesaikan suatu persamaan differensial.

Untuk memulai menggunakan `SymPy`, kita harus memanggil *library* `SymPy` terlebih dahulu. Pada tutorial kali ini, kita akan menggunakan `Jupyter notebook`, supaya hasil dari perhitungan `SymPy` bisa langsung terlihat.

```
1 import sympy
2 sympy.init_session()
3 x, y, z = symbols("x y z ")
4
5 x + z # semisal kita ingin menghitung x + z
```

Baris 1 dan 2 merupakan kode untuk memanggil *library* `SymPy`, sedangkan baris 3 merupakan perintah untuk mendefinisikan variabel dalam perhitungan. Pada contoh kasus ini, kita menggunakan variabel x, y dan z. Apabila kita ingin menghitung $x + z$, maka kita tinggal mengetikkan kode seperti pada baris 5.

Berikut ini beberapa contoh ekspresi matematik yang sering dipakai dalam perhitungan:

```
1 exp(-I*x), cos(x), atan(x), besselj(0,x), sqrt(pi)
```

Kode-kode tersebut akan menghasilkan output sebagai berikut:

$$e^{-ix}, \cos(x), \arctan(x), J_0(x), \sqrt{\pi}, \tag{6.1}$$

dimana, I merupakan bilangan imajiner.

Apabila kita ingin melakukan expansi aljabar untuk $x(z+y)^2$, maka kita dapat mengetikkan perintah berikut:

```
expand(x*(z+y)**2)
```

dengan output sebagai berikut,

$$xy^2 + 2xyz + xz^2. \tag{6.2}$$

Sebaliknya, bila kita ingin menyederhanakan suatu persamaan (misalnya kita ingin menyederhanakan Persamaan (6.2)), maka ada dua jenis perintah yang bisa digunakan, yaitu:

```
simplify(x*y**2+2*x*y*z + x*z*z)
factor(x*y**2+2*x*y*z + x*z*z)
```

Masing-masing perintah akan menghasilkan output sebagai berikut:

$$x\left(y^2 + 2yz + z^2\right) \tag{6.3}$$

dan

$$x\left(y+z\right)^2. \tag{6.4}$$

Selanjutnya, kita akan coba melakukan perhitungan simbolik untuk kasus yang lebih kompleks.

6.1 Limit, Turunan dan Integral

6.1.1 Limit

Semisal kita ingin menghitung limit dari fungsi berikut,

$$\lim_{x\to 0} \frac{\sin(x)}{x}\ , \tag{6.5}$$

maka kita tahu bahwa jawaban dari Persamaan (6.5) adalah 1. Untuk menghitung persamaan (6.5) dengan `SymPy`, kita dapat menuliskan perintah sebagai berikut:

```
limit(sin(x)/x,x,0)
```

Dan memang hasil yang kita peroleh adalah 1.

Secara umum, perintah `limit` terdiri atas tiga argumen: argumen pertama adalah fungsi yang akan dihitung limitnya; argumen kedua adalah variabel dari fungsi tersebut; dan argumen ketiga adalah batas dari limit tersebut. Mari kita mengambil contoh perhitungan limit yang lain.

$$\lim_{x\to\infty} \frac{3+4x}{5+6x}\ . \tag{6.6}$$

Perintah SymPy yang kita tulis adalah sebagai berikut:

```
limit((3+4*x)/(5+6*x),x,oo)
```

dengan output $2/3$. oo merupakan kode untuk ∞.

6.1.2 Turunan

Semisal kita ingin menghitung turunan fungsi sebagai berikut:

$$\frac{d}{dx}(5x^2 - \cos(3x) + 6xzy) \tag{6.7}$$

Maka perintah SymPy yang kita tulis adalah sebagai berikut:

```
f=5*x**2-cos(3*x)+6*x*y*z
diff(f,x,1)
```

dimana, kita mendefinisikan fungsi tersebut sebagai f. Perintah `diff` memiliki tiga argumen: argumen pertama adalah fungsi yang akan diturunkan; argumen kedua adalah variabel fungsi yang diturunkan; dan argumen ketiga adalah orde dari turunan. Dalam contoh ini kita memilih orde 1, karena kita menghitung turunan pertama. Keluarannya adalah sebagai berikut:

$$10x + 6yz + 3\sin(3x). \tag{6.8}$$

Contoh lain, semisal kita ingin menghitung turunan dari fungsi Bessel:

$$\frac{d^2}{dx^2}J_y(x) \tag{6.9}$$

dimana, y adalah orde dari fungsi Bessel. Maka perintah SymPy yang kita tulis adalah sebagai berikut:

```
f=besselj(y,x)
diff(f,x,2)
```

dengan hasil sebagai berikut,

$$\frac{-2J_y(x) + J_{y-2}(x) + J_{y+2}(x)}{4}. \tag{6.10}$$

6.1.3 Integral

Mari kita coba integralkan fungsi pada persamaan (6.8) seperti berikut:

$$\int (10x + 6yz + 3\sin(3x))\, dx. \tag{6.11}$$

Untuk menghitung intgeral tanpa batas seperti ini, input SymPy yang harus kita ketikkan adalah sebagai berikut:

```
f=10*x + 6*y*z + 3 * sin(3*x)
integrate(f,x)
```

dengan output sebagai berikut:

$$5x^2 - \cos(3x) + 6xzy. \tag{6.12}$$

Argumen pertama dari fungsi `integrate` adalah fungsi yang akan diintegralkan, sedangkan argumen kedua adalah variabel integral.

Bagaimana dengan integral yang memiliki batas seperti di bawah ini?

$$\int_a^b dx(10x + 6yz + 3\sin(3x)). \tag{6.13}$$

Untuk menyelesaikan integral tersebut, input pada SymPy adalah sebagai berikut:

```
f=10*x + 6*y*z + 3 * sin(3*x)
integrate(f,(x,a,b))
```

Contoh selanjutnya adalah intergral rangkap dua seperti di bawah ini:

$$\int_a^b dx \int_c^d dy(10x + 6yz + 3\sin(3x)). \tag{6.14}$$

Input SymPy adalah sebagai berikut:

```
f=10*x + 6*y*z + 3 * sin(3*x)
integrate(f,(x,a,b),(y,d,c))
```

Contoh terakhir adalah integral berikut yang merupakan definisi fungsi Gamma ($\Gamma(y)$):

$$\int_0^\infty dx x^{y-1} e^{-x}. \tag{6.15}$$

Maka, input SymPy yang harus diketikkan ialah sebagai berikut:

```
f=x**(y-1)*exp(-x)
integrate(f,(x,0,oo))
```

Hasilnya,

$$\Gamma(y) \qquad \text{for re}(y) - 1 > -1 \tag{6.16}$$

6.2 Ekspansi fungsi, Akar dari fungsi, dan Matriks

6.2.1 Ekspansi fungsi

Semisal kita ingin mengkespansi suatu fungsi dalam bentuk deret. Ekspansi ini dikenal dengan nama ekspansi Taylor. Sebagai contoh, semisal kita ingin mengekspansi fungsi berikut,

$$\sin(x) \tag{6.17}$$

pada titik $x = a$, maka input pada SymPy adalah sebagai berikut:

```
f=sin(x)
series(f,x,a,3)
```

Pada kasus ini, kita melakukan ekspansi sampai tiga deret. Output dari perhitungan ini adalah

$$\sin(a) + (-a+x)\cos(a) - \frac{(-a+x)^2 \sin(a)}{2} + O\left((-a+x)^3; x \to a\right) \tag{6.18}$$

dimana, O menyatakan orde error dari perhitungan. Argumen pertama dari series adalah fungsi yang ingin dideretkan, argumen kedua adalah variabel dari deret, argumen ketiga adalah titik evaluasi deret dan argumen keempat adalah jumlah orde deret yang diinginkan.

Contoh lainnya adalah ekspansi dari fungsi Bessel berikut di sekitar $x = 0$,

$$\frac{J_0(x)}{x} \tag{6.19}$$

Perintah SymPy yang harus kita ketikkan adalah sebagai berikut:

```
f=besselj(0,x)/x
series(f,x,0,3)
```

dengan hasil sebagai berikut:

$$\frac{1}{x} - \frac{x}{4} + O\left(x^3\right) \tag{6.20}$$

6.2.2 Akar dari fungsi

Akar dari suatu fungsi $f(x)$ didefinisikan sebagai berikut:

$$f(x_0) = 0 \tag{6.21}$$

dimana, x_0 adalah akar dari fungsi $f(x)$. Semisal kita ingin mencari akar dari fungsi kuadratik berikut

$$ax^2 + bx + c \tag{6.22}$$

Maka, input pada SymPy adalah sebagai berikut:

```
f=a*x**2+b*x+c
solve(f,x)
```

Hasilnya,

$$\left[\frac{-b+\sqrt{-4ac+b^2}}{2a}, -\frac{b+\sqrt{-4ac+b^2}}{2a}\right] \tag{6.23}$$

Argumen pertama dari `solve` adalah fungsi yang akan dicari akarnya dan argumen kedua adalah variabel akar. Perintah `Solve` juga dapat digunakan untuk menyelesaikan suatu persamaan transendental seperti berikut:

$$\cos(x) = \sin(2x) \tag{6.24}$$

Maka, solusi dari persamaan tersebut dapat dihitung menggunakan `SymPy` dengan perintah sebagai berikut:

```
f=cos(x)-sin(2x)
solve(f,x)
```

yang memberikan hasil untuk x sebagai berikut:

$$\left[-\frac{\pi}{2}, \frac{\pi}{6}, \frac{\pi}{2}, \frac{5\pi}{6}\right] \tag{6.25}$$

6.2.3 Matriks

Matriks pada `SymPy` dapat ditulis dengan cara seperti berikut:

```
M = Matrix([[1,2],[3,4]])
M
```

yang memberikan output matriks M sebagai berikut:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{6.26}$$

Perkalian matriks dapat dilakukan seperti berikut:

```
M = Matrix([[1,2],[3,4]])
M*M
```

yang menghasilkan keluaran berupa perkalian matriks MM

$$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix} \tag{6.27}$$

`Determinat` dan `inverse` dari suatu matriks dapat dihitung dengan perintah sebagai berikut:

```
M = Matrix([[1,2],[3,4]])
M.det()
M.inv()
```

Baris kedua akan memberikan `determinant` dari matriks M, sedangkan baris ketiga memberikan `inverse` dari matriks M. `Eigen value` dan `eigen vektor` dari matriks M dapat dihitung dengan input sebagai berikut:

```
M = Matrix([[1,2],[3,4]])
M.eigenvects()
```

Hasilnya,

$$\left[\left(\frac{5}{2}-\frac{\sqrt{33}}{2},\ 1,\ \left[\begin{bmatrix}-\frac{2}{-\frac{3}{2}+\frac{\sqrt{33}}{2}}\\ 1\end{bmatrix}\right]\right),\ \left(\frac{5}{2}+\frac{\sqrt{33}}{2},\ 1,\ \left[\begin{bmatrix}-\frac{2}{-\frac{\sqrt{33}}{2}-\frac{3}{2}}\\ 1\end{bmatrix}\right]\right)\right] \tag{6.28}$$

Ada dua eigen value dan eigen vektor dari matriks M. Kolom pertama dari masing-masing solusi adalah nilai eigen, kolom kedua adalah multiplikasi dari nilai eigen, dan kolom ketiga adalah vekto reigen.

Terakhir, kita juga bisa mendiagonalisasi matriks M. Pada kasus ini, kita menuliskan matriks M sebagai $M = PDP^{-1}$, di mana D adalah matriks yang sudah didiagonalisasi dan matriks P adalah matriks yang mendiagonalisasi matriks M. Matriks P dan D diperoleh melalui perintah `SymPy` sebagai berikut:

```
M = Matrix([[1,2],[3,4]])
M.diagonalize()
```

dengan hasil sebagai berikut,

$$\left(\begin{bmatrix}-\frac{\sqrt{33}}{6}-\frac{1}{2} & -\frac{1}{2}+\frac{\sqrt{33}}{6}\\ 1 & 1\end{bmatrix},\ \begin{bmatrix}\frac{5}{2}-\frac{\sqrt{33}}{2} & 0\\ 0 & \frac{5}{2}+\frac{\sqrt{33}}{2}\end{bmatrix}\right) \tag{6.29}$$

dimana, matriks pada kolom pertama adalah matriks P dan matriks pada kolom kedua adalah matriks D. Dapat kita lihat bahwa matriks D hanya memiliki komponen diagonal yang tidak lain adalah nilai eigen dari matriks M.

6.3 Nilai numerik dari ekspresi analitik

Sejauh ini, kita hanya mengevaluasi ekspresi analitik dari suatu perhitungan. Kini kita lakukan hal yang berbeda. Misalnya, kita ingin mengevaluasi nilai numerik dari perhitungan berikut:

$$f(x) = \frac{d^2}{dx^2}(x^3 e^{-5x}) \tag{6.30}$$

Untuk mengevaluasi nilai numerik dari fungsi $f(x)$ tersebut, perintah SymPy yang kita ketikkan adalah sebagai berikut:

```
f=lambdify(x,diff(x**3*exp(-5*x),x,2))
f(0.1)
```

Pada baris pertama, kita menggunakan perintah `lambdify` agar dapat mengevaluasi secara numerik dari suatu fungsi. Argumen pertama pada `lambdify` adalah variabel fungsi yang akan dievaluasi, sedangkan argumen kedua adalah fungsi yang akan dievaluasi. Baris kedua akan memberikan nilai numerik dari fungsi untuk $x = 0.1$. Perintah SymPy di atas akan memberikan keluaran berupa nilai numerik sebagai berikut:

$$0.19712246440660586 \tag{6.31}$$

Dengan menggunakan perintah `lambdify`, kita dapat membuat visualisasi dari ekspresi analitik yang kita peroleh dari SymPy. Perintah untuk membuat visualisasi tersebut adalah sebagai berikut:

```
import numpy as np
import matplotlib.pyplot as plt
f=lambdify(x,diff(x**3*exp(-5*x),x,2))
xx = np.linspace(0,5,500)
yy = f(xx)
plt.plot( xx,yy)
plt.tick_params( labelsize  = 18 )
plt.ylabel('f(x)',size  = 18)
plt.xlabel('x',size  = 18)
plt.show()
```

Output dari perintah di atas adalah berupa plot dari $f(x)$.

7 | Menulis Naskah dengan LaTeX

OLEH: SASFAN ARMAN WELLA

LaTeX (dibaca LAH-tekh atau LAY-tekh) merupakan *software* literasi yang digunakan untuk menulis suatu naskah (artikel, buku, dll.) dengan kualitas tinggi. Tidak seperti Ms. Word (pada Windows) atau Page (pada MacOS) yang mana *"what you see is what you get"*, LaTeX lebih cocok disebut sebagai "*software* persiapan" untuk menulis sebuah naskah, sebab apa yang kita ketikkan tidak sama dengan tampilan naskah yang dihasilkan. Apa yang kita ketik akan dieksekusi terlebih dahulu oleh sistem `TeX`[1] yang didesain oleh Donald E. Knuth.

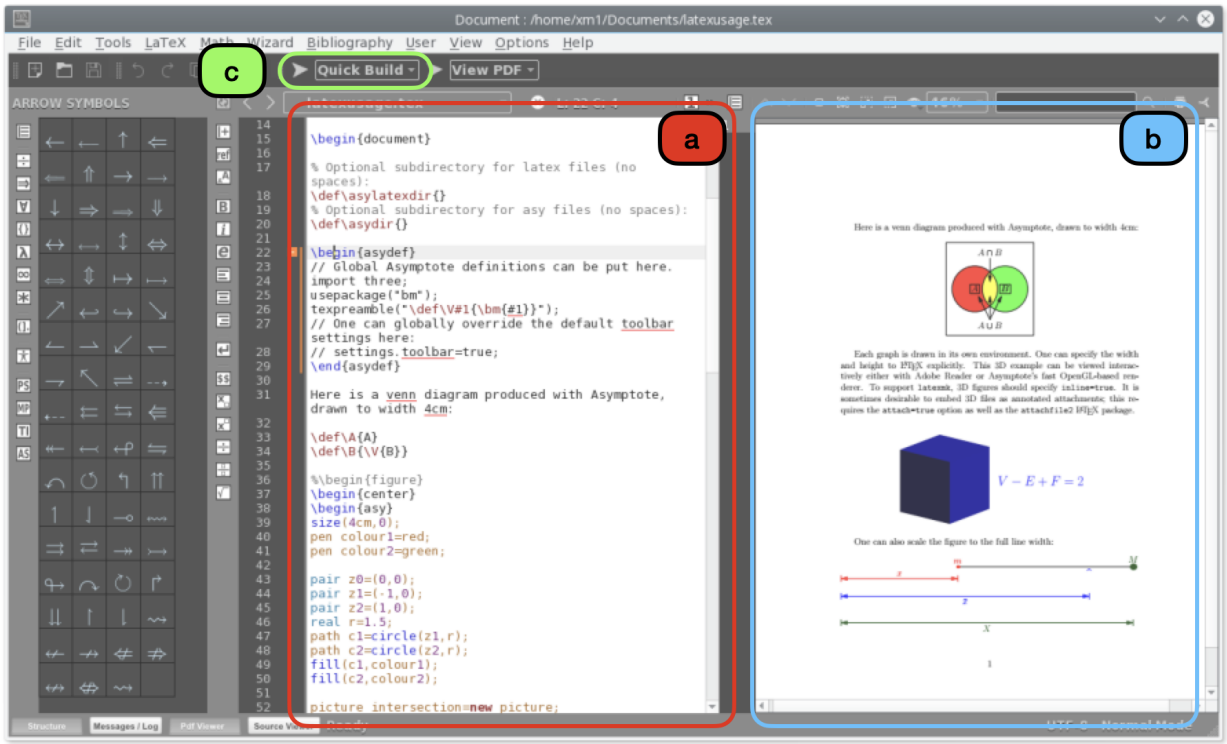
7.1 Instalasi LaTeX

7.1.1 Mode *Offline*

LaTeX bisa dijalan diberbagai sistem operasi, yakni: Windows, Linux, dan MacOS. Pada dasarnya, untuk bisa menjalankan LaTeX di komputer, kita perlu menginstal (1) sistem *engine* untuk `TeX`, dan juga (2) program editor untuk menuliskan kode-kode `TeX`. Sistem *engine* `TeX` yang biasa digunakan, yaitu `MikTeX`. *Installer* `MikTeX` tersedia untuk Windows, Linux, dan juga MacOs, dan dapat diunduh secara gratis melalui tautan berikut: https://miktex.org/download Khusus untuk MacOS, kita juga dapat menggunakan `MacTeX`[2] sebagai alternatif. Baik `MikTeX` maupun `MacTeX` berisi paket-paket atau *library* yang dibutuhkan dalam menulis naskah menggunakan LaTeX. Keduanya menyediakan jenis *installer* yang berbeda-beda tergantung pada kelengkapan *library* yang akan diinstal. Sebagai percobaan, kita bisa memilih versi *basic installer* terlebih dahulu. Tutorial lengkap mengenai instalasi `MikTeX` untuk Windows dapat dilihat pada laman berikut: https://miktex.org/howto/install-miktex, sedangkan untuk MacOS dapat dilihat di https://miktex.org/howto/install-miktex-mac.

[1] https://en.wikipedia.org/wiki/TeX
[2] http://www.tug.org/mactex

Selanjutnya, kita memerlukan program editor untuk mengetikkan konten naskah yang akan kita buat sekaligus kode-kode TeX yang kita ingin terapkan pada naskah. Apabila kita menginstal MikTeX, maka program ini menyediakan program editor bawaan, yakni TeXworks. Alternatif lainnya, kita dapat juga menggunakan Texmaker[3] atau TeXstudio[4]. Pada dasarnya, editor teks seperti: Notepad (Windows); gedit, vi, nano, atau emacs (MacOS); tetap bisa digunakan, tetapi proses kompilasi harus kita lakukan secara manual. Oleh karena itu, sangat disarankan untuk menggunakan program editor yang memang diperuntukkan untuk LaTeX.



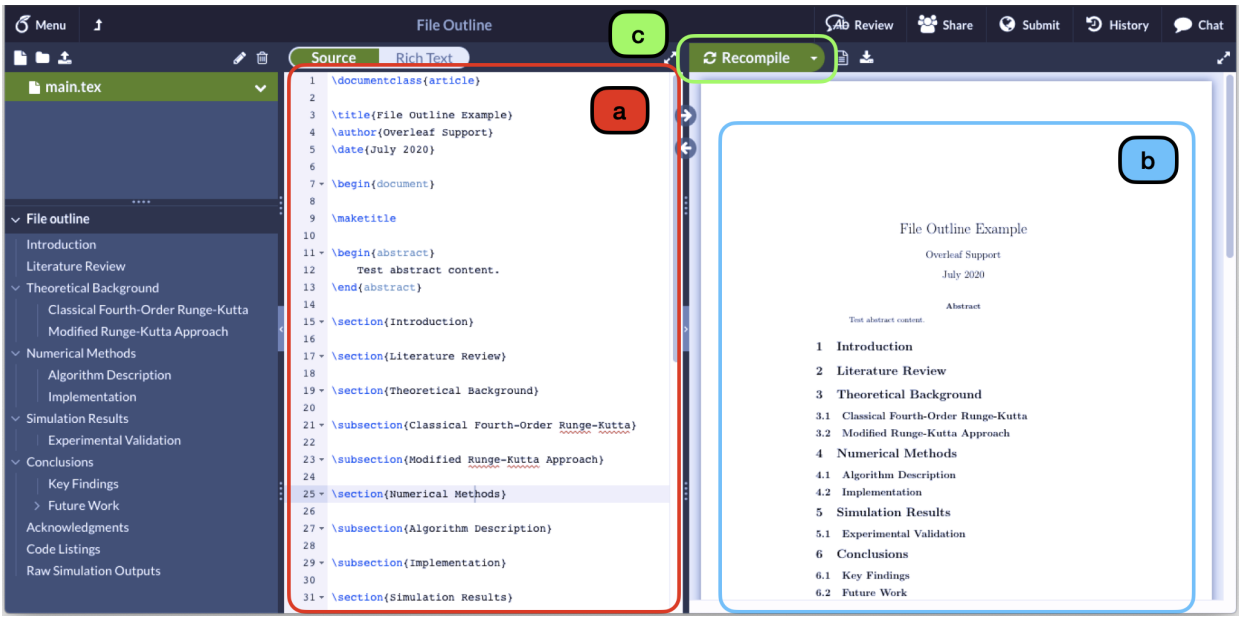
Gambar 7.1: Tampilan jendela Texmaker.

Gambar 7.1 merupakan tampilan ketika kita menggunakan Texmaker. Bagian (a) merupakan tempat dimana kita menuliskan kode-kode TeX dan juga konten naskah yang ingin kita tulis. Setelah dikompilasi dengan menekan tombol Quick Build (c), kita akan memperoleh hasil seperti ditunjukkan pada bagian (b). Dari sini kita bisa melihat bahwa penulisan naskah dengan LaTeX tidak menganut *"what you see is what you get"*. Sepintas terlihat cukup kompleks, tapi jangan khawatir, kita akan membahas penggunaan kode-kode TeX ketika menulis suatu naskah.

[3]https://www.xm1math.net/texmaker/
[4]https://www.texstudio.org/

7.1.2 Mode *Online*

Apabila instalasi di atas dirasa tidak mudah untuk dilakukan, kini kita bisa memanfaatkan editor LaTeX *online*, contohnya: `Overleaf`.[5] Lalu, bagaimana dengan sistem *engine* TeX-nya? Disini, sistem *engine* TeX sudah diintegrasikan secara lengkap. Yang perlu kita siapkan hanyalah pendaftaran akun. Keunggulan lain dari `Overleaf` ialah memungkinkan kita untuk melakukan kolaborasi, sehingga kita bisa mengerjakan satu proyek secara bersama-sama. Perlu diperhatikan bahwa untuk fitur ini, satu proyek hanya bisa dikerjakan oleh dua akun saja apabila kita menggunakan akun *free*. Untuk menambah jumlah kolaborator yang bisa mengakses proyek tersebut, kita perlu membayar sejumlah uang agar akun kita menjadi akun *pro*. Berikut ini tampilan ketika kita menggunakan `Overleaf`:



Gambar 7.2: Tampilan jendela `Overleaf`.

Jika kita bandingkan tampilan `Overleaf` ini dengan tampilan `Texmaker`, keduanya sangatlah mirip. Bagian (a), pada Gambar 7.2, merupakan tempat dimana kita menuliskan kode-kode `TeX` dan juga konten naskah yang ingin kita tulis. Bagian (b) merupakan tampilan dari *file* `.pdf` yang dihasilkan setelah kita melakukan kompilasi kode-kode di bagian (a). Untuk melakukan kompilasi tersebut kita tinggal menekan tombol `compile` atau `recompile` seperti yang ditunjukkan pada bagian (c).

[5] https://www.overleaf.com/

7.2 *Template* Standar LaTeX

Pada dasarnya, penulisan kode LaTeX terdiri atas dua bagian: (1) *Preamble* dan (2) *Document Environment*. Berikut ini merupakan *template* sederhana LaTeX:

```
1 \documentclass{article}
2
3 \usepackage{margin=2cm}{geometry}
4 \usepackage{amsmath, amssymb}
5 \usepackage{graphicx}
6
7 \begin{document}
8 % Tempat menuliskan konten naskah
9 Modul: Penulisan Naskah dengan LaTeX
10 \end{document}
```

Baris 1-5 merupakan bagian *preamble*. Baris 1 digunakan untuk menyatakan bahwa kita akan menggunakan `class` untuk pembuatan *article*. Jika kita ingin membuat buku, maka kita dapat menggunakan `class` *book*, atau jika ingin membuat *slide* presentasi, maka kita sebaiknya menggunakan `class` *beamer*. Baris 3-5 merupakan paket-paket atau *library* yang ditambahkan untuk keperluan tertentu. Apabila kita menginstal sistem *engine* `TeX` versi penuh, kita hanya tinggal memanggil paket-paket tersebut, sebab semua paket sudah tersedia. Jika kita menggunakan versi *basic*, ada kemungkinan beberapa paket tidak tersedia. Oleh karena itu, kita perlu menginstal/menambahkan paket tersebut terlebih dahulu. Baris 3 merupakan paket yang digunakan untuk mengatur margin dari halaman artikel yang kita akan buat. Pada kasus ini, kita menentapkan margin 2 cm untuk semua bagian (kiri, kanan, atas, dan bawah) halaman artikel. Baris 4 merupakan kode untuk memanggil paket `amsmath` sekaligus `amssym`. Keduanya dikembangkan oleh *American Mathematical Society* (AMS). Paket `amsmath` berisi fasilitas untuk menulis formula-formula matematis, sedangkan paket `amssym` berisi simbol-simbol matematis. Petunjuk lengkap penggunaan paket `amsmath` dapat dilihat pada tautan berikut: http://www.ams.org/arc/tex/amsmath/amsldoc.pdf. Baris 5 merupakan kode yang kita gunakan agar dapat memasukkan gambar berupa `.png` atau `.jpg` ke dalam naskah.

Baris 7-10 merupakan bagian *document environment*, tempat kita menuliskan naskah. Bagian *document environment* dimulai dengan kode `\begin{document}` dan diakhiri dengan kode `\end{document}`. Baris 8 hanya merupakan "komentar", sehingga tidak akan dieksekusi oleh *TeX engine*. Seperti yang bisa kita lihat, untuk menulis suatu "komentar" (dalam 1 baris) kita harus mengawalinya dengan tanda "`%`". Baris 9 merupakan konten naskah yang ingin kita tulis.

Apabila kode di atas dikompilasi, maka akan diperoleh hasil sebagai berikut:

Modul: Penulisan Naskah dengan LaTeX

7.3 Manipulasi teks (sederhana)

Selanjutnya, kita akan belajar bagaimana memanipulasi teks, seperti; mengubah ukuran huruf, *style* huruf.

7.3.1 Mengubah Ukuran Huruf

Untuk menetapkan ukuran huruf *default* pada naskah yang kita buat, kita dapat menetapkannya pada bagian `class` dokumen, seperti berikut:

```
\documentclass[12pt]{article}
```

Di sini, kita menetapkan ukuran huruf *default* sebesar 12 pt. Apabila, kita tidak menetapkan ukurannya secara spesifik, maka `TeX` *engine* akan menggunakan ukuran huruf 10 pt. Untuk beberapa jenis `class` yang lain (*report, letter, dll.*), ukuran *default*-nya bisa 11 atau 12 pt.

Untuk mengubah ukuran huruf untuk kata atau kalimat tertentu saja, kita bisa menggunakan kode-kode berikut:

Kode	Ukuran huruf *default* (dalam pt)		
	10	11	12
`\tiny`	5	6	6
`\scriptsize`	7	8	8
`\footnotesize`	8	9	10
`\normalsize`	10	11	12
`\large`	12	12	14
`\Large`	14	14	17
`\LARGE`	17	17	20
`\huge`	20	20	25
`\Huge`	25	25	25

Kini, mari kita coba ubah ukuran huruf dari kata "Quasi" untuk ukuran yang berbeda-beda:

```
{\large{Quasi}} {\Large{Quasi}} {\LARGE{Quasi}} {\Huge{Quasi}}
```

Hasilnya,

Quasi1 Quasi2 Quasi3 Quasi4

Penting untuk diperhatikan mengenai penggunaan tanda kurung kurawal "{}" sebelum kode ukuran dan setelah kata/kalimat yang ingin diubah ukurannya {\large{Quasi1}}. Hal ini untuk memastikan bahwa perubahan ukuran huruf yang kita lakukan, hanya berlaku untuk kata "Quasi1" saja. Jika tidak, maka perubahan ukuran huruf tersebut berlaku untuk kata/kalimat berikutnya.

7.3.2 Mengubah *Style* Huruf

Sekarang, kita akan belajar mengubah *style* huruf, seperti: menebalkan huruf (*bolt*), memiringkan huruf (*italic*), atau memberi garis bawah (*underline*).

```
\textbf{Quasi}     % Untuk menebalkan huruf
\textit{Quasi}     % Untuk memiringkan huruf
\underline{Quasi} % Untuk memberi garis bawah
```

Hasilnya,

Quasi *Quasi* Quasi

Kita juga bisa menggunakan beberapa *style* sekaligus seperti contoh berikut ini:

```
\underline{\textit{Quasi}}  % Miring & garis bawah
\textbf{\textit{Quasi}}     % Miring & tebal
\textbf{\underline{Quasi}}  % Garis bawah & tebal
```

Hasilnya,

Quasi ***Quasi*** **Quasi**

Perlu diperhatikan bahwa `\underline` tidak dapat secara otomatis melakukan pindah baris apabila sudah mencapai batas `margin` sebelah kanan, seperti kasus berikut ini:

Quasi atau QUAntum design and SImulation, Pusat Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia.

Untuk mengatasi hal tersebut, kita bisa menggunakan kode `\uline`. Untuk dapat menggunakan kode tersebut, kita perlu menambahkan paket `\usepackage{ulem}` terlebih dahulu.

Quasi atau QUAntum design and SImulation, Pusat Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia.

7.3.3 Penekanan Kata / *Emphasis*

Untuk melakukan penekanan kata pada suatu kalimat, kita bisa menggunakan kode `\emph{kata}`.

```
Quasi merupakan kependekan dari \emph{QUAntum} design and SImulation
    .
```

Quasi merupakan kependekan dari *QUAntum* design and SImulation.

Sepintas, penggunaan `\emph` serupa dengan `\textit` (*italic*). Perbedaanya, `\emph` akan menyesuaikan *style* huruf di sekitanya. Jika suatu kalimat sudah dalam dimiringkan, maka `\emph` akan membuat kata menjadi teks normal agar terlihat penekanan pada kata tersebut.

```
\textit{Quasi merupakan kependekan dari \emph{QUAntum} design and
    SImulation.}
```

Quasi merupakan kependekan dari QUAntum *design and SImulation.*

7.3.4 Jenis Huruf

Secara *default*, LaTeX akan menggunakan jenis huruf Roman. Dalam penulisan modul ini, jenis huruf yang digunakan sebagai *default* adalah jenis huruf Sans serif. Jenis huruf lain yang bisa kita gunakan ialah `Typewriter`. Berikut ini tampilan untuk ketiga jenis huruf tersebut:

```
\textrm{Quasi}
\textsf{Quasi}
\texttt{Quasi}
```

Hasilnya,

Quasi Quasi Quasi

7.3.5 Penjajaran Teks / *Alignment*

Penjajaran teks akan sangat kita butuhkan dalam penulisan naskah agar terlihat lebih rapih. Secara *default*, LaTeX menggunakan penjajaran teks `justify` atau rata kanan-kiri. Selain `justify`, kita pun bisa menggunakan mode penjajaran teks: rata kanan (*right*), rata kiti (*kiri*), dan rata tengah (*center*).

```
% Rata kanan
\begin{flushright}
A. Quasi (QUAntum design and SImulation), Pusat Penelitian Fisika,
   Lembaga Ilmu Pengetahuan Indonesia.
\end{flushright}
```

```
% Rata kiri
\begin{flushleft}
B. Quasi (QUAntum design and SImulation), Pusat Penelitian Fisika,
   Lembaga Ilmu Pengetahuan Indonesia.
\end{flushleft}
```

```
% Rata tengah
\begin{center}
C. Quasi (QUAntum design and SImulation), Pusat Penelitian Fisika,
   Lembaga Ilmu Pengetahuan Indonesia.
\end{center}
```

A. Quasi atau QUAntum design and SImulation, Pusat Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia.	B. Quasi atau QUAntum design and SImulation, Pusat Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia.	C. Quasi atau QUAntum design and SImulation, Pusat Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia.

Ganti Baris / *Line Breaks*

Ketika menggunakan Microsoft Word (Windows), untuk ganti/pindah baris, kita hanya tinggal menekan tombol `Enter/Return` pada *keyboard*. Untuk melakukan ganti/pindah baris pada LaTeX kita dapat menggunakan kode-kode berikut:

```
\begin{flushright}
Untuk rata kanan: \\
Quasi Indonesia \\[\baselineskip]
Pusat Penelitian Fisika \\[2\baselineskip]
LIPI
\end{flushright}
```

```
% Rata kiri
\begin{flushkiri}
Untuk rata kiri: \\
Quasi Indonesia \\[\baselineskip]
Pusat Penelitian Fisika \\[2\baselineskip]
LIPI
\end{flushkiri}
```

```
% Rata tengah
\begin{center}
Untuk rata tengah: \\
Quasi Indonesia \\[\baselineskip]
Pusat Penelitian Fisika \\[2\baselineskip]
LIPI
\end{center}
```

Untuk rata tengah: Quasi Indonesia Pusat Penelitian Fisika LIPI	Untuk rata tengah: Quasi Indonesia Pusat Penelitian Fisika LIPI	Untuk rata tengah: Quasi Indonesia Pusat Penelitian Fisika LIPI

Jadi, seperti yang kita lihat, kita perlu menggunakan tanda "\\" untuk pindah ke baris selanjutnya. Untuk pindah sebanyak 2 baris (1 baris dibiarkan kosong), maka kita menggunakan "\\[\baselineskip]". Untuk pindah sebanyak 3 baris (2 baris dibiarkan kosong), maka gunakan "\\[2\baselineskip]", dan seterusnya.

7.3.6 Indentasi

Ketika kita membaca suatu paragraf, kita sering melihat ada bagian yang menjorok ke dalam. Ini disebut dengan indentasi. Secara *default*, LaTeX akan memberi indentasi pada setiap paragraf, kecuali untuk paragraf pertama dari setiap BAB atau sub-BAB. Paragraf akan dibuat menjorok sejauh $1.5\times$ ukuran huruf *default*. Untuk mengatur jarak identasi, kita dapat menggunakan kode `\setlength{\parindent}{jarak}`. Apabila kita menginginkan paragraf tertentu tidak menjorok ke dalam, kita dapat menggunakan kode `\noindent` di awal paragraf. Sebagai catatan, untuk menulis paragraf baru pada LaTeX, kita hanya perlu memberikan baris kosong antara satu paragraf dengan paragraf yang lain.

Beberapa contoh identasi dapat kita lihat pada contoh berikut:

```
% Default
Quasi merupakan kependekan dari QUAntum design and SImulation
    Laboratory.
Quasi adalah kelompok penelitian yang fokus pada penelitian terkait
    teori dan komputasi material.

Quasi berada di bawah naungan Pusat Penelitian Fisika, Lembaga Ilmu
    Pengetahuan Indonesia, yang bertempat di Kawasan Puspiptek,
    Serpong, Tangerang. Selatan.
```

```
% Tanpa indentasi
Quasi merupakan kependekan dari QUAntum design and SImulation
    Laboratory.
Quasi adalah kelompok penelitian yang fokus pada penelitian terkait
    teori dan komputasi material.

```

```
\noindent Quasi berada di bawah naungan Pusat Penelitian Fisika,
    Lembaga Ilmu Pengetahuan Indonesia, yang bertempat di Kawasan
    Puspiptek, Serpong, Tangerang. Selatan.
```

```
% Dengan identasi sejauh 2 cm
Quasi merupakan kependekan dari QUAntum design and SImulation
    Laboratory.
Quasi adalah kelompok penelitian yang fokus pada penelitian terkait
    teori dan komputasi material.

\setlength{\parindent}{2cm} Quasi berada di bawah naungan Pusat
    Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia, yang
    bertempat di Kawasan Puspiptek, Serpong, Tangerang. Selatan.
```

Default

Quasi merupakan kependekan dari QUAntum design and SImulation Laboratory. Quasi adalah kelompok penelitian yang fokus pada penelitian terkait teori dan komputasi material.

Quasi berada di bawah naungan Pusat Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia, yang bertempat di Kawasan Puspiptek, Serpong, Tangerang.

Tanpa indentasi

Quasi merupakan kependekan dari QUAntum design and SImulation Laboratory. Quasi adalah kelompok penelitian yang fokus pada penelitian terkait teori dan komputasi material.
Quasi berada di bawah naungan Pusat Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia, yang bertempat di Kawasan Puspiptek, Serpong, Tangerang.

Dengan indentasi 2 cm

Quasi merupakan kependekan dari QUAntum design and SImulation Laboratory. Quasi adalah kelompok penelitian yang fokus pada penelitian terkait teori dan komputasi material.

Quasi berada di bawah naungan Pusat Penelitian Fisika, Lembaga Ilmu Pengetahuan Indonesia, yang bertempat di Kawasan Puspiptek, Serpong, Tangerang.

7.4 Notasi dan Formula Matematis

Pada naskah ilmiah, tidak jarang kita menemukan formula matematis di dalamnya. LaTeX memiliki keunggulan di bagian ini, sebab LaTeX dapat menampilkan formula matematis tersebut dengan rapih dan elegan. Kali ini, kita akan belajar untuk menuliskan notasi dan formula matematis dengan LaTeX.

7.4.1 Tampilan Formula Matematis pada LaTeX

Pada umumnya, formula matematis yang ada di dalam naskah ilmiah, bisa kita temukan pada bagian tubuh paragraf (mode *inline*) atau ditulis pada baris terpisah (mode *display*). Berikut ini dari kedua kasus tersebut:

```
% PMode \textit{inline}
Sebuah polinomial dengan satu variabel bebas dan koefisien yang
   konstan memiliki bentuk: $a_nx^n + \hdots + a_2x^2 + a_1x + 1_0$.
    Pangkat tertinggi pada suatu polinomial menunjukkan orde dari
   polinomial tersebut.

% Mode \textit{display}
Sebuah polinomial dengan satu variabel bebas dan koefisien yang
   konstan memiliki bentuk:
\begin{displaymath}
a_nx^n + \hdots + a_2x^2 + a_1x + a_0
\end{displaymath}
Pangkat tertinggi pada suatu polinomial menunjukkan orde dari
   polinomial tersebut.
```

Hasilnya,

Mode *inline*	Mode *display*
Sebuah polinomial dengan satu variabel bebas dan koefisien yang konstan memiliki bentuk: $a_nx^n + \ldots + a_2x^2 + a_1x + 1_0$. Pangkat tertinggi pada suatu polinomial menunjukkan orde dari polinomial tersebut.	Sebuah polinomial dengan satu variabel bebas dan koefisien yang konstan memiliki bentuk: $$a_nx^n + \ldots + a_2x^2 + a_1x + a_0.$$ Pangkat tertinggi pada suatu polinomial menunjukkan orde dari polinomial tersebut.

Kita dapat melihat bahwa untuk mode *inline*, kita hanya tinggal menuliskan formulasi matematis (sesuai ketentuan LaTeX) yang di awali dan diakhiri dengan tanda dollar, `${formula}$`. Untuk mode *display*, kita perlu menuliskan formula matematisnya di antara `\begin{displaymath}` dan `\end{displaymath}`.

Apabila kita ingin menampilkan formula matematis mode *display* tapi juga sekaligus melakukan penomoran pada formula tersebut, kita dapat menggunakan kode `\begin{equation}` dan `\end{equation}`.

```
% Mode display + Penomoran
```

```
Sebuah polinomial dengan satu variabel bebas dan koefisien yang
   konstan memiliki bentuk:
\begin{equation}
a_nx^n + \hdots + a_2x^2 + a_1x + a_0
\end{equation}
Pangkat tertinggi pada suatu polinomial menunjukkan orde dari
   polinomial tersebut.
```

Hasilnya,

Sebuah polinomial dengan satu variabel bebas dan koefisien yang konstan memiliki bentuk:

$$a_nx^n + \ldots + a_2x^2 + a_1x + a_0 \tag{1}$$

Pangkat tertinggi pada suatu polinomial menunjukkan orde dari polinomial tersebut.

Alternatif lainnya, kita bisa juga menggunakan kode `\begin{align}` dan `\end{align}`. Dengan kode ini, formula matematis akan ditampilkan dengan mode *display* disertai dengan penomoran di sisi sebelah kanan. Tetapi, jika kita ingin menghilangkan penomorannya, maka kita hanya tinggal menambahkan tanda bintang "$*$" menjadi `\begin{align*}` dan `\end{align*}`.

Berbeda dengan `\begin{equation}`, `\begin{align}` memiliki keunggulan lain, yakni dapat melakukan penjajaran (*alignment*) untuk persamaan yang berkelanjutan atau untuk persamaan kelompok. Contoh:

```
% Mode display + Alignment
\begin{align}
f(x)  &= a_2x^2 + a_1x + a_0 \\
      &= 2x^2 + 7x + 3
\end{align}
```

$$\begin{aligned} f(x) &= a_2x^2 + a_1x + a_0 && (1) \\ &= 2x^2 + 7x + 3 && (2) \end{aligned}$$

Kita bisa melihat bahwa persamaan (1) dan (2) merupakan persamaan berkelanjutan yang diposisikan sedemikian rupa sehingga mengalami penjajaran mulai dari tanda "$=$".

Mekanismenya, kita memisahkan formula matematisnya menjadi dua bagian: (i) bagian kiri, dan (ii) bagian kanan tanda "=". Pembagian tersebut dilakukan dengan menggunakan tanda "&" di antara kedua bagian tersebut.

Dari hasil kompilasi di atas, kita bisa melihat bahwa kedua persamaan tersebut mendapat penomoran secara otomatis. Apabila kita ingin menghapus penomoran untuk baris persamaan tertentu, kita dapat menambahkan kode `\nonumber` di akhir persamaan atau sebelum tanda ganti baris "\\".

7.4.2 Notasi Dasar

Pada bagian ini kita akan mempelajari beberapa kode untuk notasi-notasi matematik. Perlu diperhatikan bahwa kode-kode hanya bisa dieksekusi pada mode `math`. Artinya, kita perlu mengetikkannya di dalam tanda `$$`, `\begin{displaymath}`, dsb.

Operasi Aritmatika

Kode

```
a + b
a - b
a \cdot b
a \times b
a \div b
```

Output

$$a + b \tag{1}$$
$$a - b \tag{2}$$
$$a \cdot b \tag{3}$$
$$a \times b \tag{4}$$
$$a \div b \tag{5}$$

Pembagian Menggunakan Fraksi

Kode umumnya, `\frac{pembilang}{penyebut}`.

Kode

```
\dfrac{1}{2}  % mode display
\tfrac{1}{2}  % mode inline
```

Output

$$\dfrac{1}{2} \tag{1}$$
$$\tfrac{1}{2} \tag{2}$$

Superscript dan *Subscript*

Kode	Output	
`z^2`	z^2	(1)
`x_1`	x_1	(2)
`a_1^2`	a_1^2	(3)
`a^2_1`	a_1^2	(4)

Output untuk kode pada baris 3 sama dengan baris 4, walaupun kodenya berbeda. Hal ini bukan masalah, sebab yang membedakan keduanya hanyalah interpretasinya saja.

Perlu diperhatikan juga, jika *superscript* dan *subscript* lebih dari satu karakter, maka kita wajib menggunakan kurung kurawal.

Kode	Output	
`e^{kx}  % dengan tanda {}`	e^{kx}	(1)
`e^kx  % tanpa tanda {}`	$e^k x$	(2)

Superscript dan *subscript* juga dapat digunakan untuk menuliskan batas atas dan batas bawah suatu notasi penjumlahan, suatu integral, limit, dll.

Kode	Output	
`sum_{i=1}^N`	$\sum_{i=1}^N$	(1)
`\prod_p`	$\prod_p$	(2)
`\int_{x=0}^\infty`	$\int_{x=0}^\infty$	(3)
`\lim_{x \rightarrow 0}`	$\lim_{x \rightarrow 0}$	(4)

Huruf Yunani

α	\alpha	β	\beta	γ	\gamma
δ	\delta	ϵ	\epsilon	ζ	\zeta
η	\eta	θ	\theta	μ	\mu
κ	\kappa	λ	\lambda	σ	\sigma
ν	\nu	ξ	\xi	ϕ	\phi
π	\pi	ρ	\rho	ω	\omega
τ	\tau	υ	\upsilon	Θ	\Theta
χ	\chi	ψ	\psi	Π	\Pi
Γ	\Gamma	Δ	\Delta	Φ	\Phi
Λ	\Lambda	Ξ	\Xi	Ω	\Omega
Σ	\Sigma	Υ	\Upsilon	Ψ	\Psi

Bisa kita lihat, kode huruf Yunani untuk *lowercase* dan *uppercase* hanya berbeda pada huruf pertamanya saja, contoh: \delta (δ) dan \Delta (Δ).

Penggunaan Tanda Kurung

Mari kita bandingkan kedua kode berikut:

a) Tanda kurung biasa

```
[ \sum_{x=0}^N ( \frac{1}{1+x} )^x ]^2
```

b) Dengan \left dan \right

```
\left[ \sum_{x=0}^N \left( \frac{1}{1+x} \right)^x \right]^2
```

Hasilnya,

a) Tanda kurung biasa

$$[\sum_{x=0}^{N}(\frac{1}{1+x})^x]^2$$

b) Dengan \left dan \right

$$\left[\sum_{x=0}^{N}\left(\frac{1}{1+x}\right)^x\right]^2$$

Terlihat bahwa penggunaan \left dan \right menghasilkan tampilan yang lebih baik. Jika menggunakan tanda kurung biasa, baik tanda ( ) maupun [ ] memiliki ukuran yang sama, tidak peduli ukuran formula di dalamnya. Dengan menggunakan \left dan \right, selain ukuran tanda kurung mengikuti ukuran formula di dalamnya, tapi juga menyesuaikan dengan tingkatan dari tanda kurung tersebut. Dalam kasus ini, tanda [ ] posisinya lebih luar daripada ( ), sehingga ukuran [ ] lebih besar dari ( ).

Kita sebenarnya bisa mengatur ukuran tanda kurung secara manual. Tetapi, cara ini tidak mempedulikan tingkatan tanda kurung di dalam formula dan juga tidak memperhatikan ukuran formula di dalamnya.

```
(Q) \big( Q \big) \Big( Q \Big) \bigg( Q \bigg) \Bigg( Q \Bigg)
```

$$(Q) \quad \big(Q \big) \quad \Big(Q \Big) \quad \bigg(Q \bigg) \quad \Bigg(Q \Bigg)$$

Fungsi Spesial

Perhatikan kedua kode ini:

a) Kode biasa

```
sin(x) cos(x) tan(x) exp(x) quasi(x)
```

b) Dengan fungsi khusus

```
\sin(x) \cos(x) \tan(x) \exp(x) \operatorname{quasi}(x)
```

Hasilnya,

a) Kode biasa	b) Dengan fungsi khusus
$sin(x)\ cos(x)\ tan(x)\ exp(x)\ quasi(x)$	$\sin(x)\ \cos(x)\ \tan(x)\ \exp(x)\ \operatorname{quasi}(x)$

Terlihat bahwa dengan menggunakan fungsi-fungsi tersebut, kita memperoleh tampilan yang lebih baik. Nama fungsi ditulis dengan mode `teks` sedangkan variabel ditulis dengan mode `math`. Untuk fungsi yang tidak disediakan oleh LaTeX, kita dapat menggunakan kode:

$$\backslash\texttt{operatorname}\{\texttt{fungsi}\}\{\texttt{variabel}\}.$$

Integral

Kode	Output	
`\int`	$\int$	(1)
`\iint`	$\iint$	(2)
`\iiint`	$\iiint$	(3)
`\oint`	$\oint$	(4)

Turunan

Kode	Output	
`\frac{\mathrm{d} Q}{\mathrm{d }x}`	$\frac{\mathrm{d} Q}{\mathrm{d }x}$	(1)
`\frac{\partial Q}{\partial x}`	$\frac{\partial Q}{\partial x}$	(2)
`\dot{x}(t)`	$\dot{x}(t)$	(3)
`\ddot{x}(t)`	$\ddot{x}(t)$	(4)

Notasi Lainnya

Kode	Output	
`\vec{r}(t)  % Notasi vektor`	$\vec{r}(t)$	(1)
`\vv{r}(t)  % Perlu \usepackage{esvect}`	$\vec{r}t$	(2)
`\hat{H}  % Notasi operator`	$\hat{H}$	(3)
`\sqrt{x}  % Akar pangkat 2`	$\sqrt{x}$	(4)
`\sqrt[3]{y}  % Akar pangkat 2`	$\sqrt[3]{y}$	(5)
`\pm  % plus minus`	$\pm$	(6)
`\mp  % minus plus`	$\mp$	(7)

Spasi pada Notasi Matematik

Kode	Output
`a b % spasi`	ab (1)
`a \, b % backslash + koma`	$a\,b$ (2)
`a \: b % backslash + titik dua`	$a\:b$ (3)
`a \; b % backslash + titik koma`	$a\;b$ (4)
`a \quad b % backslash + 'quad'`	$a \quad b$ (5)

Matriks

Kali ini kita akan belajar untuk menulis suatu matriks dengan LaTeX. Pada dasarnya, kode penulisan matriks adalah sebagai berikut:

```
\begin{ array }{ alignment }
...
\end{ array }
```

Bagian `alignment` merupakan tempat kita menentukan penjajaran untuk tiap elemen matriks dalam kolom tertentu: `c` untuk rata tengah, `l` untuk rata kiri, dan `r` untuk rata kanan. Untuk memberikan garis vertikal antar kolom, kita tinggal menambahkan tanda | pada posisi dimana kita ingin memberikan garis. Sedangkan untuk memberikan garis horizontal, kita menggunakan kode `\hline` di bagian (...). Hal lain yang wajib diperhatikan, setiap elemen matriks dalam satu baris harus dipisahkan dengan tanda &. Beberapa contoh penulisan matriks dapat dilihat pada tabel di bawah ini:

Kode	Output

```
\begin{array}{ccc}
  a_{11}  & a_{12}  & a_{13}  \\
  a_{21}  & a_{22}  & a_{23}  \\
  a_{31}  & a_{32}  & a_{33}
\end{array}
```

$$\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}$$

```
\begin{array}{c | cc}
  a_{11}  & a_{12}  & a_{13}  \\
  a_{21}  & a_{22}  & a_{23}  \\
  a_{31}  & a_{32}  & a_{33}
\end{array}
```

$$\begin{array}{c|cc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}$$

```
\begin{array}{ccc}
  a_{11}  & a_{12}  & a_{13}  \\
  \hline
  a_{21}  & a_{22}  & a_{23}  \\
  a_{31}  & a_{32}  & a_{33}
\end{array}
```

$$\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ \hline a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}$$

Kode	Output

```
\begin{pmatrix}
  a_{11}  & a_{12}  & a_{13}  \\
  a_{21}  & a_{22}  & a_{23}  \\
  a_{31}  & a_{32}  & a_{33}
\end{pmatrix}
```

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

```
\begin{bmatrix}
  a_{11}  & a_{12}  & a_{13}  \\
  a_{21}  & a_{22}  & a_{23}  \\
  a_{31}  & a_{32}  & a_{33}
\end{bmatrix}
```

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

7.4.3 Label pada Formula Matematis

Perhatikan contoh berikut:

$$x + y = 3 \tag{1}$$
$$2x - y = 4.5 \tag{2}$$

Dengan mensubsitusi Persamaan (1) ke Persamaan (2), kita akan dapatkan nilai x dan y, masing-masing 2.5 dan 0.5.

LaTeX menyediakan fitur khusus yang sangat membantu kita untuk melakukan penulisan seperti contoh di atas. Dengan fitur ini, kita tidak perlu repot untuk mereferensikan nomor suatu persamaan secara manual. Akan lebih merepotkan, apabila terjadi perubahan susunan penomoran. Tapi dengan fitur ini, pereferensian persamaan akan berubah secara otomatis. sistem akan menyesuaikan perubahan susunan tersebut.

Mekanisme yang perlu kita lakukan untuk bisa menggunakan fitur ini, yaitu: setiap persamaan yang akan dimunculkan pada *body text* harus diberi label menggunakan kode `\label{katakunci}`. Kemudian, pada bagian *body text* kita hanya tinggal memanggil persamaan tertentu dengan kode `\ref{katakunci}`.

Untuk mendapatkan tampilan seperti contoh di atas, kita dapat menuliskan kode berikut:

```
{\begin{align}
x+y   & = 3     \label{Pers1}   \\
2x-y  & = 4.5   \label{Pers2}   \\
\end{align}}

Dengan mensubsitusi Persamaan (\ref{Pers1}) ke Persamaan (\ref{Pers2
   }), kita akan dapatkan nilai $x$ dan $y$, masing-masing 2.5 dan
   0.5.
```

7.5 Membuat Tabel

7.5.1 Perbedaan Blok `Tabular` dan `Table`

Pada dasarnya, untuk membuat tabel dengan LaTeX, kita hanya perlu menggunakan blok `tabular` dengan struktur sebagai berikut:

```
\begin{ tabular }{ alignment }
...
\end{ tabular }
```

Mekanisme pembuatan tabel dengan blok tabular ini serupa dengan pembuatan matriks yang telah kita bahas sebelumnya. Bagian `alignment` merupakan tempat kita menentukan jumlah kolom yang akan kita buat sekaligus menentukan penjajaran untuk tiap elemen dalam kolom tabel tertentu: `c` untuk rata tengah, `l` untuk rata kiri, dan `r` untuk rata kanan. Misalkan kita ingin membuat tabel 3 kolom, yang mana kolom 1-2 dijajarkan ke kanan dan kolom 3 dijajarkan ke tengah, maka pada bagian `alignment` kita tuliskan {`c c r`}. Untuk memberikan garis vertikal antar kolom, kita tinggal menambahkan tanda | pada posisi dimana kita ingin memberikan garis. Misalkan kita hanya ingin memberi gari vertikal di antara kolom 2 dan 3, maka kita tuliskan {`c c | r`}. Untuk memberikan garis horizontal, kita menggunakan kode `\hline` di bagian (...). Hal lain yang wajib diperhatikan, setiap elemen tabel dalam satu baris harus dipisahkan dengan tanda &.

Lalu apa bedanya dengan blok `table`? Blok ini perlu kita gunakan agar kita bisa memberi label dan *caption* pada tabel yang telah kita buat dengan blok `tabular`. Selain itu, dengan blok ini kita bisa menentukan penempatan tabel pada naskah, misalkan di bagian, di bawah, atau di tengah halaman naskah.

Berikut ini struktur kode ketika kita menggunakan blok `table`:

```
\begin{ table }[ placement ]
\caption{...}
\begin{ tabular }[ alignment ]
...
\end{ tabular }
\label{...}
\end{ table }
```

Untuk penempatan tabel yang kita buat, kita bisa menuliskan kode-kode berikut:

- `t` Meminta sistem untuk menempatkan tabel di bagian atas halaman naskah,
- `b` Meminta sistem untuk menempatkan tabel di bagian bawah halaman naskah,

- `h` *Here*, meminta sistem untuk menempatkan tabel sesuai dengan posisi pembuatan tabel pada *script* LaTeXyang dibuat,

- `h!` Memaksa sistem untuk sebisa mungkin menempatkan tabel sesuai *script* LaTeX.

Sebagai tambahan, kita pun bisa memberikan beberapa opsi penempatan, sebagai contoh [`tb`] artinya kita meminta sistem untuk menempatkan tabel di bagian atas atau di bagian bawah naskah.

Berikut beberapa contoh kode-kode pembuatan tabel beserta hasil keluarannya:

Kode	Output

```
\begin{tabular}{l r c}
\hline
Nama          & No. & Posisi \\
\hline
G. Buffon     & 1   & GK   \\
N. Vidic      & 15  & CB   \\
N. Kante      & 13  & MC   \\
A. Shevchenko & 7   & ST   \\
\hline
\end{tabular}
```

Nama	No.	Posisi
G. Buffon	1	GK
N. Vidic	15	CB
N. Kante	13	MC
A. Shevchenko	7	ST

```
\begin{tabular}{ | l | r | c | }
\hline
Nama          & No. & Posisi \\
\hline
G. Buffon     & 1   & GK   \\
N. Vidic      & 15  & CB   \\
N. Kante      & 13  & MC   \\
A. Shevchenko & 7   & ST   \\
\hline
\end{tabular}
```

Nama	No.	Posisi
G. Buffon	1	GK
N. Vidic	15	CB
N. Kante	13	MC
A. Shevchenko	7	ST

Kode | Output

```
\begin{table}[h!]
\caption{Daftar Pemain}
\begin{tabular}{ | l | r | c | }
\hline
Nama          & No. & Posisi  \\
\hline
G. Buffon     & 1 & GK  \\
N. Vidic      & 15  & CB  \\
N. Kante      & 13  & MC  \\
A. Shevchenko & 7 & ST  \\
\hline
\end{tabular}
\end{table}
```

Table 1. Daftar Pemain

Nama	No.	Posisi
G. Buffon	1	GK
N. Vidic	15	CB
N. Kante	13	MC
A. Shevchenko	7	ST

Catatan: Jika kita ingin mengubah tulisan `Table` pada *caption* menjadi `Tabel`, kita dapat menambahkan kode berikut pada bagian *preamble*: `\renewcommand{\tablename}{Tabel}`.

7.5.2 Penggabungan Sel (*Merge*) Horizontal

Untuk melakukan penggabungan sel (*merge*) secara horizontal, kita dapat menggunakan kode berikut: `\multicolumn{Jumlah Kolom}{Alignment}{Konten}`

Berikut ini contoh penggunaannya:

Kode | Output

```
\begin{tabular}{l r c}
\hline
Nama  &  \multicolumn{2}{ c | }{Data}
    \\
\hline
G. Buffon     & 1   & GK  \\
N. Vidic      & 15  & CB  \\
N. Kante      & 13  & MC  \\
A. Shevchenko & 7   & ST  \\
\hline
\end{tabular}
```

Nama	Data	
G. Buffon	1	GK
N. Vidic	15	CB
N. Kante	13	MC
A. Shevchenko	7	ST

Dari contoh di atas, kita bisa melihat bahwa sekarang sel `No.` dan `Posisi` digabungkan menjadi `Data`. Perhatikan juga bahwa kita menggunakan `{ c | }` pada `alignment` sel yang akan digabungkan. Tanda | perlu ditambahkan, sebab jika tidak bagian tepi sebelah kanan sel tersebut tidak mempunyai garis vertikal, seperti berikut:

Nama	Data	
G. Buffon	1	GK
N. Vidic	15	CB
N. Kante	13	MC
A. Shevchenko	7	ST

7.5.3 Penggabungan Sel (*Merge*) Vertikal

Untuk melakukan penggabungan sel (*merge*) secara vertikal, kita perlu paket khusus, yakni `\usepackage{multirow}`. Setelah selesai menambahkan paket tersebut, kita bisa menggunakan kode berikut: `\multirow{Jumlah Baris}{Lebar Sel}{Konten}` Khusus untuk bagian `Lebar Sel`, kita dapat mengisinya dengan "*" agar lebar selnya ditentukan secara otomatis.

Kode

```
\begin{tabular}{c l c}
\hline
\multirow{4}{*}{Data} & G. Buffon   &  1  \\
  & N. Vidic       & 15  \\
  & N. Kante       & 13  \\
  & A. Shevchenko & 7   \\
\hline
\end{tabular}
```

Output

Data	G. Buffon	1
	N. Vidic	15
	N. Kante	13
	A. Shevchenko	7

7.6 Menyisipkan Gambar

Kita perlu menggunakan paket `\usepackage{graphicx}`, untuk dapat menyisipkan gambar di dalam naskah. Kode yang kita tuliskan untuk menyisipkan gambar tersebut memiliki struktur sebagai berikut:

`\includegraphics[Ukuran Gambar]{Posisi File}`

Namun, agar dapat memberikan *caption* dan juga *label* pada gambar tersebut, kita perlu meletakkan kode tersebut di dalam blok `figure`, seperti berikut:

```
\begin{ figure }[ placement ]
\centering
\includegraphics[Ukuran Gambar]{Posisi File}
\caption{...}
\label{...}
\end{ figure }
```

Bagian `placement` diisi dengan kode penempatan yang mekanisme sama persis dengan tabel. Untuk bagian `Ukuran Gambar`, kita bisa mengatur seberapa besar ukuran tampilan gambar pada naskah. Sebagai contoh, kita bisa menggunakan kode

```
[width = 0.7\textwidth].
```

Artinya, gambar yang akan ditampilkan pada naskah berukuran 70% dari lebar dokumen.

Berikut ini contoh penggunaannya:

Kode	Output
`\includegraphics[width=0.3\textwidth]{./Gambar/Logo-Quasi.png}`	

`\begin{figure}[h!]`
`\centering`
`\includegraphics[width=0.3\textwidth]{./Gambar/Logo-Quasi.png}`
`\caption{Logo QUASI}`
`\end{figure}`
	
	Figure 1. Logo QUASI

Catatan: Jika kita ingin mengubah tulisan `Figure` pada *caption* menjadi `Gambar`, kita dapat menambahkan kode berikut pada bagian *preamble*: `\renewcommand{\figurename}{Gambar}`.

7.7 Membuat Daftar Isi

Untuk membuat daftar isi dengan LaTeX sangatlah mudah, kita tinggal mengetikkan kode berikut:

```
\tableofcontents.
```

Jika kita ingin mengubah judul daftar isi dari `Contents` menjadi `Daftar Isi`, kita dapat menambahkan kode berikut pada bagian *preamble*:

```
\renewcommand{\contentsname}{Daftar Isi}.
```

7.8 Membuat Daftar Pustaka

Untuk membuat daftar pustaka dengan LaTeX, kita mempunyai dua cara: (i) Menulis pustaka langsung di dalam *file* utama, dan (ii) menulis pustaka *file* terpisah (dengan *file* BibTeX).

7.8.1 Langsung pada *File* Utama

Untuk menuliskan pustaka dengan cara ini, kita perlu mengetikkan kode berikut ini di bagian paling bawah sebelum `\end{document}`:[6]

```
\begin{ thebibliography }{ X }
...
\end{ thebibliography }
```

Nilai `X` menyatakan jumlah digit dari penomoran referensi yang digunakan. Jika hanya terdapat 1-9 referensi saja, artinya penomoran bisa dilakukan dengan satu digit saja. Jadi, nilai `X` bisa diisi dengan angka 9. Jika jumlah referensinya 10-99, maka kita memerlukan dua digit penomoran. Untuk kasus ini, nilai `X` harus diisi dengan angka 99.

Selanjutnya, di bawah kode tersebut, kita mulai mengetikkan referensi-referensi yang kita gunakan dengan cara mengetikkan kode sebagai berikut:

```
\bibitem{Label} Diikuti data mengenai referensi.
```

Berikut ini contoh pembuatan daftar pustaka yang ditulis langsung pada *file* utama:

[6] https://www.overleaf.com/learn/latex/Bibliography_management_with_bibtex

```
\begin{document}

\section{First section}

This document is an example of \texttt{thebibliography} environment
   using in bibliography management. Three items are cited: \textit{
   The \LaTeX\ Companion} book \cite{latexcompanion}, the Einstein
   journal paper \cite{einstein}, and the Donald Knuth's website \
   cite{knuthwebsite}. The \LaTeX\ related items are\cite{
   latexcompanion,knuthwebsite}.

\medskip

\begin{thebibliography}{9}

\bibitem{latexcompanion} Michel Goossens, Frank Mittelbach, and
   Alexander Samarin. \textit{The \LaTeX\ Companion}. Addison-Wesley
   , Reading, Massachusetts, 1993.

\bibitem{einstein} Albert Einstein. \textit{Zur Elektrodynamik
   bewegter K{\"o}rper}. (German) [\textit{On the electrodynamics of
    moving bodies}]. Annalen der Physik, 332 (10) : 891 - 921, 1905.


\bibitem{knuthwebsite} Knuth: Computers and Typesetting,
\\\texttt{http://www-cs-faculty.stanford.edu/
\~{}uno/abcde.html}


\end{thebibliography}

\end{document}
```

Hasilnya,

1 First section

This document is an example of **thebibliography** environment using in bibliography management. Three items are cited: *The LaTeX Companion* book [1], the Einstein journal paper [2], and the Donald Knuth's website [3]. The LaTeX related items are [1, 3].

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[2] Albert Einstein. *Zur Elektrodynamik bewegter Körper*. (German) [*On the electrodynamics of moving bodies*]. Annalen der Physik, 322(10):891–921, 1905.

[3] Knuth: Computers and Typesetting, http://www-cs-faculty.stanford.edu/~uno/abcde.html

Untuk melakukan sitasi untuk ketiga referensi di atas, kita bisa menggunakan kode `\cite{Label}`, contoh: `\cite{einstein}` untuk mensitasi referensi kedua.

7.8.2 Dengan Menggunakan BibTeX

Jika kita ingin membuat daftar pustaka dengan menggunakan BibTeX, pertama, kita perlu mengetikkan dua kode berikut sebelum `\end{document}`:

```
\bibliographystyle{ Nama style }
\bibliography{ Nama file .bib }
```

Kode pertama digunakan untuk menetapkan format penulisan pustaka. Ada beberapa *style* yang disediakan oleh LaTeX, antara lain: `unsrt`, `abbrv`, `plain`, dsb. Informasi lengkap mengenai *style* pustaka dapat dilihat melalui tautan berikut: https://www.overleaf.com/learn/latex/Bibtex_bibliography_styles.

Kode kedua merupakan nama file `.bib` yang berisi data mengenai pustaka yang kita akan gunakan. Data masing-masing pustaka harus ditulis dalam format BibTeX (`contoh.bib`) seperti berikut:

```
@article{einstein,
    author =        "Albert Einstein",
```

```
    title =         "{Zur Elektrodynamik bewegter K{\"o}rper}. ({German})
        [{On} the electrodynamics of moving bodies]",
    journal =       "Annalen der Physik",
    volume =        "322",
    number =        "10",
    pages =         "891--921",
    year =          "1905",
    DOI =           "http://dx.doi.org/10.1002/andp.19053221004"
}

@book{latexcompanion,
    author    = "Michel Goossens and Frank Mittelbach and Alexander Samarin",
    title     = "The \LaTeX\ Companion",
    year      = "1993",
    publisher = "Addison-Wesley",
    address   = "Reading, Massachusetts"
}

@misc{knuthwebsite,
    author    = "Donald Knuth",
    title     = "Knuth: Computers and Typesetting",
    url       = "http://www-cs-faculty.stanford.edu/\~{}uno/abcde.html"
}
```

Berikut ini contoh penulisan pustaka menggunakan BibTeX:

```
\begin{document}

This document is an example of BibTeX using in bibliography management. Three items are cited: \textit{The \LaTeX\ Companion} book \cite{latexcompanion}, the Einstein journal paper \cite{einstein}, and the Donald Knuth's website \cite{knuthwebsite}. The \LaTeX\ related items are \cite{latexcompanion,knuthwebsite}.

\medskip

\bibliographystyle{unsrt}
\bibliography{contoh}

\end{document}
```

DIsini kita menggunakan *style* `unsrt`. Dan kita meminta sistem membaca data mengenai pustaka melalui *file* `contoh.bib`. Apabila kita kompilasi, maka hasilnya akan sama dengan output ketika kita membuat daftar pustaka secara langsung pada *file* utama.

Keunggulan dari cara ini, kita umumnya tidak perlu mengetikkan data-data mengenai pustaka yang kita ingin sitasi, sebab setiap jurnal menyediakan *file* BibTeX untuk naskah artikel atau buku yang kita akan sitasi.

- - - o O o - - -