KALKULASI NILAI EIGEN BERBASIS MATRIKS HAMILTONIAN MENGGUNAKAN TEKNIK *BLOCK MATRIX* STUDI KASUS PADA *GRAPHENE*

SKRIPSI

Diajukan sebagai Salah Satu Syarat untuk Menempuh Ujian Akhir Tingkat Sarjana
pada Program Studi Fisika Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Padjadjaran

Oleh:
Mirza Aditya Deliantama
140310170044

PROGRAM STUDI FISIKA
FAKULTAS MATEMATIKA & ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJAJARAN
2021

LEMBAR PENGESAHAN

JUDUL	:	KALKULASI NILAI EIGEN BERBASIS MATRIKS HAMILTONIAN MENGGUNAKAN TEKNIK *BLOCK MATRIX* : STUDI KASUS PADA *GRAPHENE*
PENYUSUN	:	MIRZA ADITYA DELIANTAMA
NPM	:	140310170044
LAB	:	FISIKA INSTRUMENTASI

Jatinangor, Juli 2021
Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
Prof. Dr. I Made Joni, M.Sc.	Dr. Ferry Faizal, Ph.D.
NIP. 19720601 200112 1 001	NIP. 19820531 201903 3 001

Mengetahui,
Ketua Program Studi Fisika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Padjadjaran

Dr. Andri Abdurrochman, S.Si, MT.
NIP. 19740526 200312 1 002

KATA PENGANTAR

Dengan nama Allah yang Maha Pengasih dan Maha Penyayang. Alhamdulillaahirabbil'aalamiin, segala puji hanya milik Allah 'azza wa jalla karena berkat-Nya skripsi yang berjudul "Kalkulasi nilai eigen berbasis matriks Hamiltonian menggunakan Teknik *Block Matrix* : studi kasus pada *Graphene*" sebagai salah satu syarat menyelesaikan pendidikan tingkat sarjana pada Program Studi Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Padjadjaran dapat selesai. Tak lupa terimakasih kepada Drs. Enung Achmad S., M.Pd., dan Lilis Susilayati, S.H., yaitu orang tua penulis yang selalu memberi dukungan kepada penulis sehingga penulis dapat menyelesaikan dengan sebaik mungkin. Penulis juga mengucapkan terimakasih kepada pihak-pihak yang membantu penulis untuk menyelesaikan tugas akhir ini:

1. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran, Prof. Dr. Iman Rahayu, M.Sc.

2. Kepala Departemen Fisika FMIPA Unpad, Prof. Dr. Camellia Panatarani

3. Ketua Program Studi Fisika FMIPA Unpad, Dr. Andri Abdurrochman, S.Si, MT.

4. Prof. Dr. Eng. I Made Joni, M.Sc., selaku pembimbing utama dan dosen wali yang telah yang selalu memberikan bimbingan, dukungan, bantuan secara

materi dan psikis dan fasilitas terbaik kepada penulis selama penelitian dan kuliah

5. Prof. Dr. Eng Camellia Panatarani, M.Si., yang selalu memberikan ilmu, dukungan dan fasilitas terbaik kepada penulis selama menyelesaikan skripsi
6. Ferry Faizal, Ph.D., selaku pembimbing pendamping yang selalu memberikan bimbingan, saran dan masukan, bantuan, dukungan, fasilitas dan arahan terbaiknya kepada penulis
7. Dr. Wildan Abdussalam, yang selalu memberikan bimbingan, motivasi, bantuan, dan arahan yang membuat penulis selalu bersemangat saat menyelesaikan skripsi
8. Pa Setianto, Pa Ocik, Kang Dwindra, dan seluruh civitas FiNder-U CoE yang membantu penulis menyelesaikan skripsi
9. Nurfauzi Fadillah yang telah memberikan bantuan, dukungan, fasilitas dan motivasi kepada penulis
10. P-Labs sebagai tempat penulis untuk membuat, merevisi dan menyelesaikan skripsi dengan menyenangkan
11. Ferdian, Syakir, Yoga, Felix, Hilmy, Ahmad, Victor, Naufal, Ghemma, Fahmi, Bayu, Reyhan, Budi, dan Urfan yang telah memberikan kebahagiaan, dukungan dan bantuan kepada penulis selama menyelesaikan skripsi

12. Revani dan Fahriza yang telah memberikan dukungan dan bantuan saat mengadakan pemaparan kemajuan mingguan agar skripsi ini dapat selesai dengan baik
13. Lutfi Naufal Ramadhika dan Lucia Patia Rochman yang telah memberikan dukungan moril kepada penulis saat menyelesaikan skripsi
14. Atom 2017 yang telah mewarnai masa kuliah penulis
15. Tim KBK Instrumentasi dan Elektronika yang selalu membuat penulis bersemangat untuk mengerjakan skripsi

Dengan segala kerendahan hati, penulis menyadari bahwa skripsi ini masih memiliki banyak kekurangan dan jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat dibutuhkan oleh penulis. Semoga skripsi ini dapat bermanfaat bagi perkembangan ilmu dan teknologi serta bagi pihak-pihak yang membutuhkan.

Jatinangor, Juli 2021

Penulis

ABSTRAK

Senyawa kimia organik adalah salah satu senyawa kimia yg sering diteliti, salah satu cara untuk meneliti senyawa tersebut adalah dengan menggunakan pendekatan kuantum yg disebut Kimia Fisik. Graphene merupakan salah satu senyawa kimia organik yang dapat diamati karakteristiknya dengan menggunakan Teori Hückel, yang akan melihat interaksi antar ikatan-π dalam senyawa tersebut. Penggunaan kalkulasi eigen dari persamaan Schrödinger tak bergantung waktu akan menghasilkan nilai energi orbital molekul dan nilai konstanta gelombang. Kedua nilai inilah yang akan digunakan untuk mencari nilai absorbsi. Ketika menggunakan matriks Hamiltonian yang ukurannya besar, maka akan menyebabkan adanya *bottleneck* dalam memproses data. Agar performa komputasinya tidak menurun, digunakan perhitungan menggunakan teknik *Block Matrix* dan teknik *Big Data*. Hasil penelitian menunjukkan bahwa, semakin banyak atom karbon yang dimiliki Graphene akan membuat celah energi yang dihasilkan semakin kecil yang diakibatkan oleh menurunnya jarak antara energi orbital molekul. Selain itu dihasilkan juga performa komputasi tercepat untuk menghitung celah energi, yaitu dengan menggunakan PySpark. Nilai perubahan eksitasi energi dan momen dipol transisi yang didapatkan dari kalkulasi nilai eigen akan menggambarkan grafik absorpsi terhadap energi eksitasinya, dimana titik absorbsi tertinggi menggambarkan titik celah energinya.

Kata kunci: ***Graphene, Teori Hückel, Teori Pita, Energi Absorbsi, PySpark***

ABSTRACT

Organic chemical compounds are one of the chemical compounds that are often studied, one way to investigate these compounds is to use a quantum approach called Physycochemical. Graphene is one of the organic chemical compounds whose characteristics can be observed using the Hückel Theory, which will look at the interactions between π-bond in these compounds. The use of the eigen calculation of the time-independent Schrödinger equation will yield the energy values of the molecular orbitals and the value of the wave constant. These two values will be used to find the absorption value. When using a larger Hamiltonian matrix, it will cause a bottleneck in processing the data. So that the computational performance does not decrease, calculations using the Block Matrix technique and the Big Data technique are used. The results show that, the more carbon atoms that Graphene has, the smaller the energy gap produced due to the decrease in the distance between the molecular orbital energies. Moreover, the fastest computational performance for calculating the energy gap is also generated, namely by using PySpark. The value of the change in excitation energy and the transition dipole moment obtained from the calculation of the eigenvalues will describe the absorption graph of the excitation energy, where the highest absorption point describes the energy gap point.

Keywords: ***Graphene, Hückel Theory, Band Theory, Absorbance Energy, PySpark***

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kimia fisik adalah subdisiplin yang membahas tentang suatu kejadian physicochemical (fisiko-kimia) dengan menggunakan pendekatan secara atomik dan molekul dan benda terkondensasi dalam kacamata kefisikaan [1]. Karena kimia fisik, maka akan sangat-sangat berkaitan dengan unsur yang ada di tabel periodik. Mulai dari unsur Hidrogen (H) dengan nomor atom 1, hingga unsur Oganeson (Og) dengan nomor atom 118. Salah satu bidang tentang kimia yaitu Kimia Organik. Kimia organik mempelajari tentang struktur, sifat, komposisi, reaksi, dan preparasi senyawa yang mengandung unsur karbon (C). Kimia organik tidak hanya mempelajari tentang hidrokarbon saja, namun ikatan antara karbon dengan senyawa lain seperti hidrogen, nitrogen, oksigen, sulfur, fosfor, silikon dan halogen [2]. Salah satu senyawa kimia organik yang saat ini sedang diteliti yaitu graphene. Graphene merupakan salah satu senyawa kimia turunan dari grafit dengan penyusun unsurnya adalah karbon yang saling berikatan satu sama lain [3]. Graphene merupakan bentuk lembaran (sheet) dari grafit. Karena molekul atom C dari graphene terdelokasi, maka kita bisa menghitung nilai energi dari orbital molekul graphene ini. Caranya adalah dengan menggunakan Teori Orbital Molekul Hückel atau sering disebut sebagai Metode Hückel [4].

Teori OMH ini memiliki karakteristik yaitu menggunakan operasi matriks untuk perhitungannnya. Jika matriks yang digunakan masih berukuran kecil, maka dapat dilakukan dengan menggunakan operasi determinan biasa. Namun karena studi kasus yang dilakukan menggunakan graphene dengan jumlah atom karbon yang sangat banyak, maka besar matriks hamiltoniannya akan membesar dan menyebabkan operasi matriks yang digunakan adalah operasi eigen. Untuk dapat melakukan perhitungan besar energi orbital molekul maka akan dilakukan secara komputasi. Namun, besarnya ukuran matriks memengaruhi kecepatan komputasi untuk mencari solusi nilai eigen dari matriks tersebut. Ini berlaku untuk beberapa platform komputasi yang sering kita gunakan seperti MATLAB, Python dan lainnya.

Untuk menjawab permasalahan tersebut, maka diperlukannya operasi menggunakan sistem paralelisasi partisi. Dengan menggunakan paralelisasi partisi, kita bisa memanipulasi atau membagi sebuah matriks besar menjadi beberapa sub matriks kecil dan tetap mempertahankan kaidah matriks seperti biasanya [5]. Pekerjaan tiap sub matriks kecil tersebut akan dilakukan secara paralel di setiap core dari processor yang digunakan. Hal ini dapat mempercepat komputasi karena tidak perlu adanya waktu tunggu (latency) karena menunggu operasi sebelumnya [6].

Karena matriks dianggap suatu data, dan ukurannya sangat besar, maka akan sangat erat kaitannya dengan Big Data. Big Data singkatnya merupakan kumpulan data yang berukuran besar. Big Data memiliki beberapa karakteristik yang

terangkum dalam “6V’s of Big Data”. Karakteristik yang terpenting yaitu tentang volume karena ukurannya yang besar dan velocity karena memiliki kecepatan komputasi yang sangat cepat mendekati waktu nyata [7]. Maka dari itu, akan dilakukan operasi matriks menggunakan sistem Block Matrix dengan komputasi Big Data.

1.2 Identifikasi Masalah

1. Bagaimana pita energi yang dimiliki oleh Graphene
2. Bagaimana pengaruh penggunaan PySpark terhadap waktu komputasi matriks yang dibutuhkan
3. Bagaimana Spektrum Absorbansi terhadap Perubahan Energi dari Graphene

1.3 Batasan Masalah

1. Memerhatikan asumsi Hückel, yaitu:
   (a) Menganggap semua ikatan adalah ikatan tunggal
   (b) Hanya berinteraksi dengan tetangga terdekat (*First-Nearest Neighbor*)
   (c) Elektron dalam ikatan-π ’merasakan’ potensial elektrostatik diakibatkan oleh seluruh ikatan-σ
   (d) Hanya melibatkan orbital elektron-π dan mengabaikan interaksi orbital elektron-σ dengan orbital elektron π

2. Menggunakan aplikasi *Apache Spark* dengan bahasa pemrograman *Python* dalam *PySpark*

3. Senyawa yang digunakan dalam perhitungan yaitu Graphene

4. Menggunakan dua buah *environment* server untuk kalkulasi:

   (a) Google Cloud Instance Iowa (US)

   (b) Server KST

1.4 Tujuan Penelitian

1. Merancang algoritma untuk membangun matriks Hamiltonian dan kalkulasi spektrum absorbsi

2. Dapat mengetahui hubungan antara besar celah energi tiap banyaknya atom yang terlibat

3. Membandingkan performa komputasi menggunakan PySpark, dan Python dengan variasi *serial processing*, *multiprocessing* dan *threading*

4. Mengetahui spektrum pita energi yang dimiliki oleh Graphene

1.5 Manfaat Penelitian

1. Mengetahui sifat dan karakteristik, khususnya tentang energi orbital molekul dari Graphene

2. Menambah pengetahuan tentang Graphene

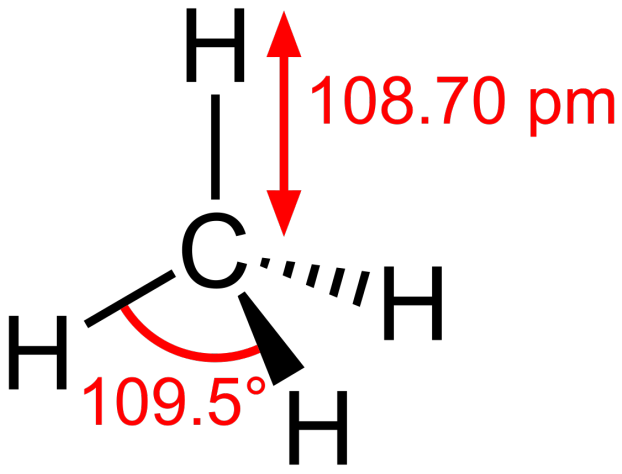
3. Mengetahui teknik pemrosesan matriks dengan laju komputasi tercepat

BAB II

TINJAUAN PUSTAKA

2.1 Kimia Organik

Kimia organik merupakan bidang kimia yang mempelajari tentang struktur, sifat, komposisi, reaksi, dan preparasi senyawa yang mengandung unsur karbon (C). Kimia organik tidak hanya tentang hidrokarbon, namun mempelajari senyawa lain juga misal unsur karbon berikatan dengan unsur hidrogen (H), nitrogen (N), oksigen (O), halogen (golongan VII A), fosfor (P), silikon (Si) dan belerang (S) [2]. Kimia organik akan memerhatikan tentang sifat fisik dan kimia beserta evaluasi reaktivitas kimia untuk memahami perilaku dari senyawa tersebut.



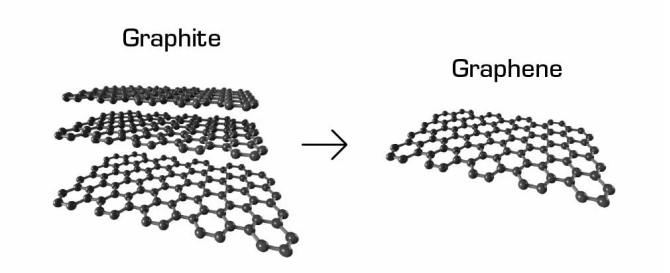
Gambar 2.1: *Senyawa Metana* [8]

Senyawa organik memiliki pola ikatan tunggal dan rangkap (rangkap dua dan tiga) karena karbon memiliki 4 elektron valensi yang menyebabkan karbon dapat

berikatan dengan 4 buah unsur yang berbeda [6]. Kimia organik sangatlah berguna untuk produk komersial dan juga produk sains. Seperti contohnya untuk kosmetik, pelumas, bahan petrokimia dan lainnya. Salah satu senyawa kimia organik yang saat ini sedang diteliti yaitu senyawa Graphene.

2.2 Graphene

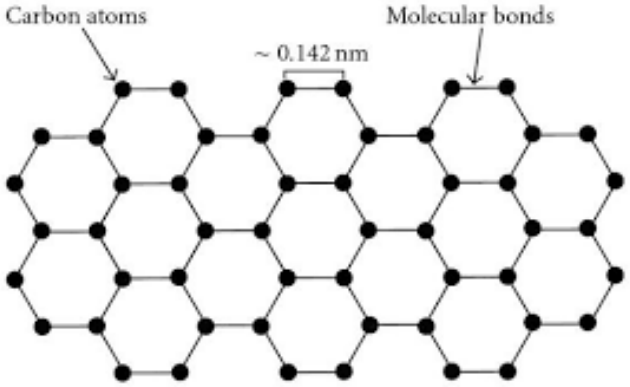
Graphene merupakan senyawa dengan ketebalan seperti ketebalan satu buah atom (*one-atom-thick*) dan berstruktur 2 dimensi yang terdiri dari miliaran atom karbon (C) yang terbentuk seperti kisi heksagonal (atau sering disebut dengan sarang lebah) dan merupakan struktur fundamental untuk setiap bentuk karbon saat ini [7][9]. Senyawa graphene memiliki karakteristik tebal yang sangat tipis, fleksibel, kuat dan transparan [5]. Graphene lebih keras daripada intan, namun lebih elastis daripada karet, lebih kuat daripada baja dan lebih ringan daripada aluminium [10]. Graphene merupakan turunan dari karbon yang dijadikan lembaran dan berasal dari graphite.



Gambar 2.2: *Struktur Graphene dan Graphite* [11]

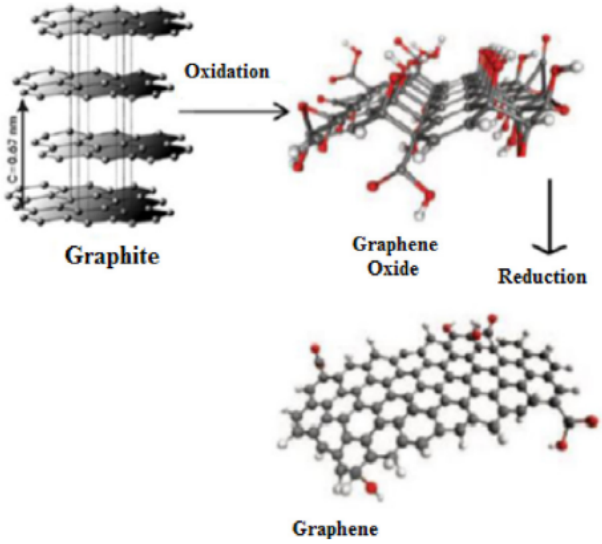
Graphene hanya terdiri dari atom C yang jarak antar atom C sebesar ± 0.142 nm [3] dan jarak interplanar nya 0.335 nm. Setiap atom karbon (atom C) memiliki

radius $\pm$ 170 pm atau 0.17 nm (*Van der Walls radius*) [12]. Graphene memiliki mobilitas elektron lebih dari 15000 $cm^2V^{-1}s^{-1}$. Dari sini bisa disimpulkan bahwa Graphene memiliki sifat kimia dan fisiknya yang sangat baik. Lalu karena Graphene adalah dasar dari alotrop grafit, lalu dapat dibentuk menjadi fullerene 0D, digulung menjadi tabung nano (nanotube) 1D, dan ditumpuk menjadi grafit 3D, inilah sebabnya Graphene disebut sebagai induk dari grafit [13].



Gambar 2.3: *Struktur Senyawa Graphene* [7]

Aplikasi dari Graphene bisa digunakan menjadi resonator, saklar, katup, konduktor dan beberapa aplikasi lainnya [7].



Gambar 2.4: *Pembuatan Graphene dari Graphite* [14]

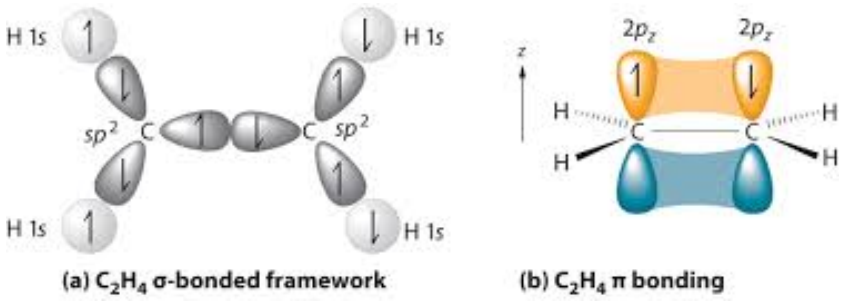
Karakteristik dari Graphene bisa kita lihat, salah satunya mengenai energi yang dimiliki oleh orbital molekul Graphene. Untuk melihat energinya, kita dapat menggunakan suatu teori yang dikemukakan oleh Enrich Hückel yang bernama Teori Hückel.

2.3 Teori Hückel

Teori Hückel sangat erat kaitannya dengan teori orbital molekul. Orbital molekul merupakan suatu penggambaran daerah yang dapat ditempati oleh suatu elektron dalam suatu molekul. Orbital molekul menggunakan pendekatan Schrödinger untuk elektron dalam medan listrik di inti atom suatu molekul. Dengan menggunakan orbital molekul kita bisa melihat konfigurasi elektron molekul. Karena orbital molekul merupakan fermion, maka ia akan memenuhi prinsip Pauli [15]. Orbital dari suatu molekul memiliki level energinya

masing-masing. Ikatan antar unsur dalam suatu molekul disebut ikatan-σ. Misalkan, senyawa C_2H_4 (etilen), memiliki ikatan C=C dengan ikatan ganda, dan C-H dengan ikatan tunggal. Ikatan C=C dan C-H disebut ikatan-σ, dan karena hibridisasi dari etana memiliki elektron di $2p_x$ dan $2p_y$, maka orbital di $2p_z$ akan membentuk ikatan-π. Karena jarak antara ikatan-π dan ikatan-σ cukup jauh, menyebabkan interaksi antar ikatan π lebih besar daripada besar interaksi ikatan σ dan π. Maka kita bisa mengabaikan interaksi antara ikatan σ dan π [16]

.



Gambar 2.5: *Orbital Molekul Etilen (C_2H_4)* [17]

Dari sini, Hückel mengembangkan suatu teori dimana orbital molekul yang digunakan dapat dihitung energinya menggunakan teori elektron-π. Maka lahirlah Teori Orbital Molekul Hückel (Teori OMH) atau sering disebut dengan Teori Hückel. Dengan menggunakan teori Hückel, kita bisa melihat energi yang dihasilkan dari ikatan-π suatu molekul. Ia mengungkapkan bahwa orbital molekul yang dilambangkan oleh ψ adalah kombinasi linier dari orbital-orbital $2p_z$ dari semua atom karbon dalam suatu molekul [18]. Kombinasi linier orbital molekul sering disebut sebagai LCAO (*Linear Combination of Atomic Orbital*) [19]. LCAO

$2p_z$ ini dapat ditulis secara matematis dengan:

$$\psi = \sum_i c_i \phi_i \tag{2.1}$$

dengan ϕ_i adalah orbital $2p_z$ dalam atom karbon ke-i. Jika $\hat{H}$ dianggap sebagai Hamiltonian efektif elektron-tunggal dalam molekul, maka berlaku:

$$\hat{H}\psi = \varepsilon\psi \tag{2.2}$$

Persamaan (2.2) memenuhi persamaan sekuler:

$$\sum_j H_{ij} - \varepsilon S_{ij} c_j = 0 \tag{2.3}$$

dengan

$$H_{ij} = \int \phi_i \hat{H} \phi_j \, dv \tag{2.4}$$

$$S_{ij} = \int \phi_i \phi_j \, dv \tag{2.5}$$

Integral yang ada di persamaan (2.4) bisa didefinisikan sebagai data empiris. Contohnya adalah H_{ii} adalah potensial ionisasi elektron-π di karbon ke-i dan $H_{i,i\pm1}$ dalah energi yang dibutuhkan untuk elektron-π melompat ke atom terdekatnya.

Karena $S_{ii} = 1$ dan S_{ij} jauh lebih kecil daripada 1 maka dapat diabaikan. Maka,

$$H_{ij} = \begin{cases} \alpha; i = j \\ \beta; j = i \pm 1 \\ 0; lainnya \end{cases} \tag{2.6}$$

Singkatnya, persamaan (2.6) dapat menjadi sebuah persamaan matriks yang ditulis dengan,

$$\begin{bmatrix} H_{11} - ES_{11} & H_{12} - ES_{12} & \dots & H_{1j} - ES_{1j} \\ H_{21} - ES_{21} & H_{22} - ES_{22} & \dots & H_{2j} - ES_{2j} \\ \dots & \dots & \dots & \dots \\ H_{i1} - ES_{i1} & H_{i2} - ES_{i2} & \dots & H_{ij} - ES_{ij} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_j \end{bmatrix} = 0 \tag{2.7}$$

Karena $H_{ii} = \alpha$, $H_{ij} = \beta$, dan $S_{ij} = \delta_{ij} = 1$ jika $i = j$ dan akan bernilai 0 jika $i! = j$. Maka matriks Hamiltoniannya menjadi,

$$\begin{bmatrix} \alpha - E & \beta & 0 & \beta \\ \beta & \alpha - E & \beta & 0 \\ \dots & \dots & \dots & \dots \\ \beta & 0 & \beta & \alpha - E \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_j \end{bmatrix} = 0 \tag{2.8}$$

Misalkan pada *Ethylene* memiliki 2 atom C yang menyebabkan matriks

Hamiltoniannya menjadi ukuran berukuran 2x2.

$$\begin{bmatrix} \alpha - E & \beta \\ \beta & \alpha - E \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = 0 \tag{2.9}$$

Energi orbital molekul dalam teori OMH didefinisikan dalam dua bentuk, yaitu energi dari sebuah elektron dalam orbital 2p yang disebut dengan α dan energi interaksi antara dua orbital 2p yang disebut dengan β [17][18].

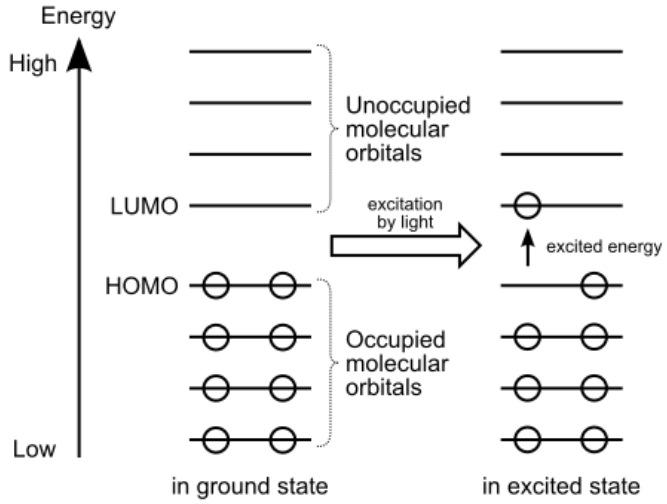
Dalam kasus ini, $\hat{H}$ merupakan suatu matriks Hamiltonian yang menggambarkan keadaan orbital dari suatu molekul. Banyaknya atom karbon akan mempengaruhi besar matriks Hamiltonian yang didapat. Seperti misalkan benzene (C_6H_6) yang memiliki 6 atom karbon, maka matriks yang dihasilkannya berukuran 6x6. Jika dikaitkan dengan graphene yang memiliki jumlah atom C yang sangat banyak, maka matriks Hamiltonian yang dihasilkannya pun ukurannya akan sangat besar. Operasi yang akan digunakan yaitu operasi eigen. Hasil dari metode ini adalah energi orbital dari molekul uji yang digunakan. Aplikasi dari metode Hückel ini yaitu bisa menghitung celah pita [20], rapat elektron dan konduktivitas bahan [16]. Dari literatur, dikatakan bahwa pada Graphene besar α = -10.7 eV dan β = -1.58 eV [5].

Sesuai persamaan Schrödinger tak bergantung waktu pada Persamaan (2.2), maka kita bisa memperlakukan persamaan tersebut sebagai persamaan eigen. $\hat{H}$ sebagai operator, $|\psi\rangle$ sebagai fungsi gelombang, dan ε sebagai nilai eigen (*eigen*

energy). Tiap tingkatan eigen energy nya akan merepresentasikan keadaan energi dalam tiap orbital yang dimiliki oleh material tertentu. Sebelum menentukan lebar celah dari tingkat energi nya, perlu untuk memahami tentang teori pita terlebih dahulu.

2.4 Teori Pita

Dalam teori fisika zat padat, saat sebuah atom digabungkan dengan atom lainnya akan terjadi tumpang tindih fungsi gelombang elektron. Misalkan terdapat atom A dan B yang digabungkan. Jika tidak ada interaksi *bonding* dan *anti-bonding* nya, maka orbital yang dihasilkan adalah non ikatan, dimana elektron yang menempati orbital molekul terisi dan berenergi tinggi disebut *Highest Occupied Molecular Orbital* (HOMO), dan orbital molekul tidak terisi dan berenergi paling rendah disebut *Lowest Unoccupied Molecular Orbital* (LUMO) [21].



Gambar 2.6: *Diagram HOMO dan LUMO pada suatu molekul* [22]

Teori pita adalah suatu model yang menjelaskan keadaan sebuah elektron yang hanya memiliki sebuah keadaan energi spesifik tertentu [17]. Dalam sebuah senyawa, elektron akan menempati suatu tempat sesuai energi spesifik yang dimilikinya. Tempat inilah yang akan disebut sebagai pita energi. Sesuai dengan larangan Pauli (*Pauli Exclusion*) dimana tidak ada dua elektron yang memiliki bilangan kuantum yang sama. Jadi jika terdapat elektron lain, elektron tersebut akan membuat pita baru dan tidak akan menempati keadaan pita yang lainnya [23].

2.5 Spektrum Absorbansi

Pada Gambar 2.6, saat terdapat cahaya berenergi tertentu ditembakkan, maka orbital yang berada di daerah HOMO akan memiliki energi berlebih. Akibatnya, orbital tersebut akan tereksitasi ke keadaan dengan energi yang lebih tinggi. Karena orbital tersebut tidak stabil saat berada di keadaan energi yang bukan keadaan asalnya, orbital tersebut akan de-eksitasi kembali ke keadaan awalnya dengan meng-emisikan energi berlebih yang ia serap menjadi suatu gelombang emisi [24]. Dari fenomena tersebut, kita bisa mengetahui celah pita yang dimiliki oleh senyawa tertentu dan bisa melihat energi yang di serap di tiap celah energinya dengan menggambarkan spektrum absorbansinya. Spektrum Absorbansi mengacu pada pengukuran penyerapan radiasi (melalui spektroskopi) sebagai fungsi energi atau panjang gelombang pada sebuah sampel. Untuk mencari absorbansi terhadap perubahan energi eksitasinya dapat menggunakan persamaan,

$$\tilde{G}(E) = \frac{1}{N}\sum_{i}^{N} |\tilde{G}^{(j)}|^2 \delta_{E,E_j} \tag{2.10}$$

dengan

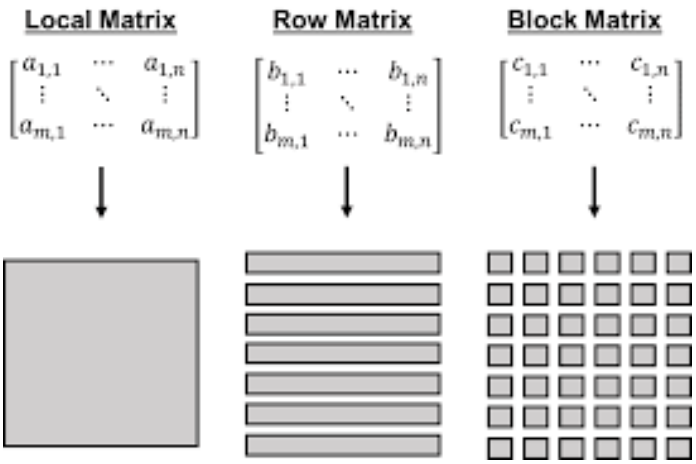
$$\tilde{G}^{(j)} = g\sum_{i}^{N} \langle \varphi_0 | \hat{\sigma}_{eg}^{(i)} + \hat{\sigma}_{ge}^{(i)} | \varphi_j \rangle \tag{2.11}$$

δ_{E,E_j} adalah delta *kronecker* dan E_j adalah *eigen energy* pada keadaan kolektif [25].

Dari penjelasan sebelumnya, permasalahan yang dihadapi yaitu ukuran matriks Hamiltonian yang sangat besar dan ini berefek pada laju komputasi yang akan kita lakukan. Hal tersebut bisa membuat laju komputasi menurun drastis karena proses komputasi yang sangat banyak dan tidak bisa dilakukan secara paralel. Untuk menjawab permasalahan tersebut, maka kita akan menggunakan operasi Block Matrix sebagai paralelisasi operasi matriks agar dapat dikerjakan dengan worker yang berbeda dan membantu dalam komputasi.

2.6 *Block Matrix*

Block Matrix merupakan salah satu operasi yang paling penting untuk meningkatkan kinerja komputasi. Dengan menggunakan Block Matrix maka operasi dapat diparalelisasi [26]. Block Matrix adalah operasi matriks dimana suatu matriks berdimensi besar akan dibagi menjadi beberapa sub matriks kecil [27].



Gambar 2.7: *Perbandingan Local Matrix, Row Matrix dan Block Matrix* [28]

Terlihat dari Gambar 2.7 bahwa Block Matrix akan membagi matriks yang awalnya berukuran sangat besar (yaitu Local Matrix), menjadi beberapa sub matriks kecil. Misalkan kita memiliki matriks A dengan ukuran 4x4,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{2.12}$$

dengan menggunakan Block Matrix, maka matriks A dapat kita bagi menjadi 4 sub matriks yaitu $A_{11}, A_{12}, A_{21}, A_{22}$.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{2.13}$$

jika ditulis per submatriksnya menjadi,

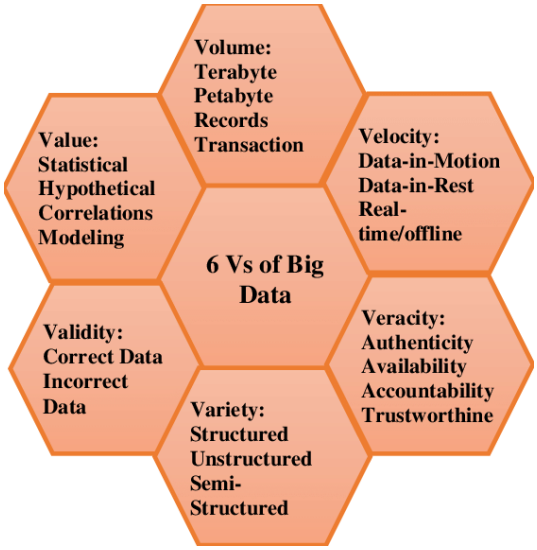
$$A_{11} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} A_{12} = \begin{bmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{bmatrix} A_{21} = \begin{bmatrix} a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} A_{22} = \begin{bmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{bmatrix} \quad (2.14)$$

Salah satu metode Block Matrix adalah dengan menggunakan *Schur Decomposition*. Dekomposisi Schur akan membagi sebuah matriks menjadi 4 buah sub matriks dan menghitung determinan tiap sub matriksnya. Jika menggunakan persamaan (2.12), maka dekomposisi Schur nya menjadi

$$det \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = det(A_{11}) * det(A_{22} - A_{21}A_{11}^{-1}A_{12}) \quad (2.15)$$

Karena elemen matriks yang didapatkan dari Graphene sangat banyak, dan diperlukan adanya sistem komputasi yang bisa memproses banyak data dengan kecepatan yang sangat tinggi. Maka kita akan menggunakan suatu metode menggunakan Big Data.

2.7 *Big Data*



Gambar 2.8: *Big Data dan Karakteristiknya* [29]

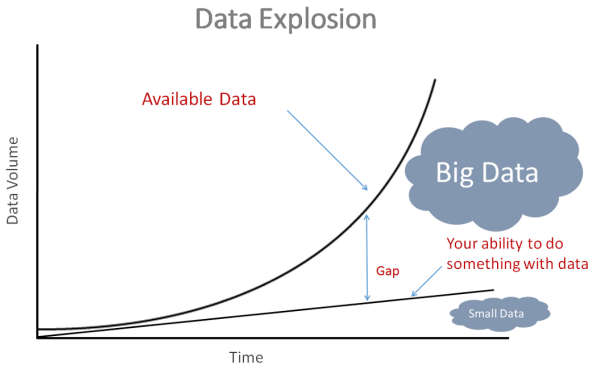
Big Data adalah salah satu metode cara memproses berbagai jenis data atau dalam literatur lain dikatakan bahwa *Big Data* adalah data yang berisi lebih banyak variasi yang datang dalam volume yang terus meningkat dan dengan kecepatan meningkat juga [30]. Data dibagi menjadi dua tipe, yaitu Data Terstruktur (data numerik), dan Data Tidak Terstruktur (teks, suara, gambar, dll) [31]. Secara makna, Big Data merupakan suatu data yang ukurannya sangat besar, bentuk yang tidak teratur, dengan komunikasi data kecepatan tinggi. Big Data memiliki 6 karakteristik yang disebut sebagai "*6V's of Big Data*" yang tertera pada Tabel 2.1.

Big Data biasa disebut sebagai Not-Only Structured Query Language (NoSQL), hal ini berbeda dengan Relational Database Management System (RDBMS) atau basis data tradisional mulai dari variasi data untuk NoSQL bisa mencakup data semi terstruktur dan data tidak terstruktur, kecepatan baca-tulis (Read-Write) data

Tabel 2.1: Karakteristik Big Data [32][33]

Volume	Volume adalah salah satu hal yang menjadi feature dari Big Data. Berkaitan dengan hubungan antara ukuran dan kapasitas pemrosesan.
Velocity	Kecepatan pemrosesan Big Data sangatlah cepat, ini dikarenakan dari data yang dimilikinya sangat besar dan akan semakin membesar seiring banyaknya data yang masuk. Kecepatan yang dimiliki sangat cepat karena memanfaatkan data geolokasi, tren dan informasi input data.
Value	Value menjelaskan bahwa nilai yang akan diperoleh dari suatu data dan lebih bernilai dari stored data
Variability	Variability mengungkap tentang variable-variabel yang mempengaruhi proses komunikasi data dan pemrosesannya beserta kemungkinan terbaiknya.
Veracity	Veracity menunjukkan kualitas dan asal data, dan sebagai penyeleksian kebenaran dan keaslian data.
Variety	Variety menggambarkan bahwa didalam Big Data terdapat banyak variasi atau macam data yang akan diproses dan dianalisis. Bise berupa data numerik, audio video, bahkan teks.

dari Big Data yang sangat cepat (bergantung pada jumlah node) dan scalable [34].



Gambar 2.9: *Grafik Ledakan Data terhadap Waktu Komputasi* [35]

Beberapa kelebihan dan kekurangan jika menggunakan Big Data terdapat pada Tabel 2.2.

Terdapat beberapa *platform* yang sering digunakan dalam pengolahan Big Data,

Tabel 2.2: Kelebihan dan Kekurangan Big Data [36]

Kelebihan	Kekurangan
Pengambilan keputusan yang lebih baik	Kebutuhan akan *talent* yang profesional di bidangnya
Peningkatan produktivitas	Kualitas data
Mengurangi biaya	Perlunya perubahan budaya untuk menggunakan Big Data
Peningkatan layanan pelanggan	Peraturan yang mengatur tentang privasi data
Deteksi penipuan	Resiko keamanan
Peningkatan pendapatan	Perubahan yang cepat
Peningkatan agility	Kebutuhan perangkat keras pendukung
Inovasi yang lebih besar	Kecepatan lebih cepat dalam pemrosesannya
Kecepatan lebih cepat dalam pemrosesannya	Biaya

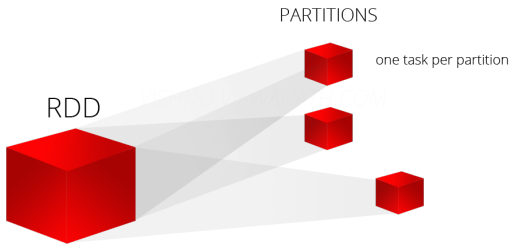
dan beberapa dan yang akan digunakannya adalah Apache Spark.

2.7.1 *Apache Spark*

Gambar 2.10: *Apache Spark* [37]

Apache Spark merupakan sebuah aplikasi mesin analitik yang terpadu untuk pemrosesan data berskala besar [37]. Apache Spark banyak digunakan di berbagai industri, seperti Netflix, Yahoo, dan eBay. Bahasa yang dapat digunakan dalam Apache Spark adalah *Scala*, *SQL* dan *Python*. Arsitektur data dari Spark berupa RDD (*Resilient Distributed Dataset*). *Resilient* berarti toleran terhadap kesalahan

dan dapat membuat data baru dari kesalahan tersebut, *Distributed* berarti data didistribusikan di antara beberapa node dalam sebuah cluster, dan *Dataset* berarti Kumpulan data yang berisi suatu nilai [38].



Gambar 2.11: *Contoh Kasus RDD pada Spark* [39]

Dengan menggunakan RDD, kita bisa melakukan beberapa hal seperti:

1. *Transformations* = Operasi ini dapat digunakan untuk membuat RDD baru
2. *Actions* = Operasi ini diaplikasikan pada RDD untuk menginstruksikan Apache Spark untuk menerapkan komputasi dan meneruskan hasilnya kembali ke driver

Kelebihan dari Spark adalah:

1. Kecepatan

   Spark memiliki kecepatan 100x lebih cepat daripada Hadoop. Ini karena Spark menggunakan komputasi dan beberapa pengoptimalan lainnya.

2. Kemudahan dalam penggunaan

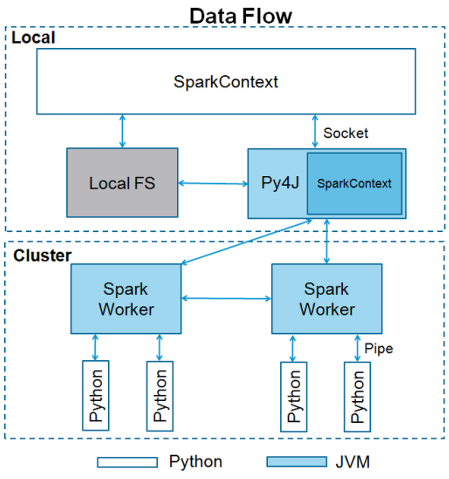
   Spark memiliki antarmuka yang mudah digunakan untuk memproses Big

Data.

3. Aplikasi terpadu

   Spark memiliki dukungan dan library yang sangat banyak, termasuk SQL, streaming data, machine learning, dan pemrosesan grafik. Salah satunya, spark dapat dioperasikan menggunakan library dari python yang disebut sebagai PySpark [40].

Apache Spark sangatlah cepat dalam pemrosesan data dikarenakan dalam kerjanya menggunakan *In-Memory Processing* yang berarti semua pekerjaan atau pemrosesan data yang dilakukan oleh Apache Spark akan di proses di dalam *memory*[41]. Untuk menggunakan fungsionalitas dari Spark, diperlukan *SparkContext* sebagai titik masuk aplikasinya. Untuk skema alur data nya dapat dilihat di Gambar 2.11 [42].



Gambar 2.12: *Alur Kerja Data pada Apache Spark* [42]

Kegunaan Apache Spark atau lebih tepatnya PySpark pada penelitian ini adalah untuk memproses komputasi energi orbital dari Teori Hückel. Basis bahasa dari

PySpark adalah Python, namun karena karakteristik yang dimiliki PySpark yang akan membuat komputasi nya akan semakin cepat dan dibandingkan dengan laju komputasi menggunakan Python biasa.

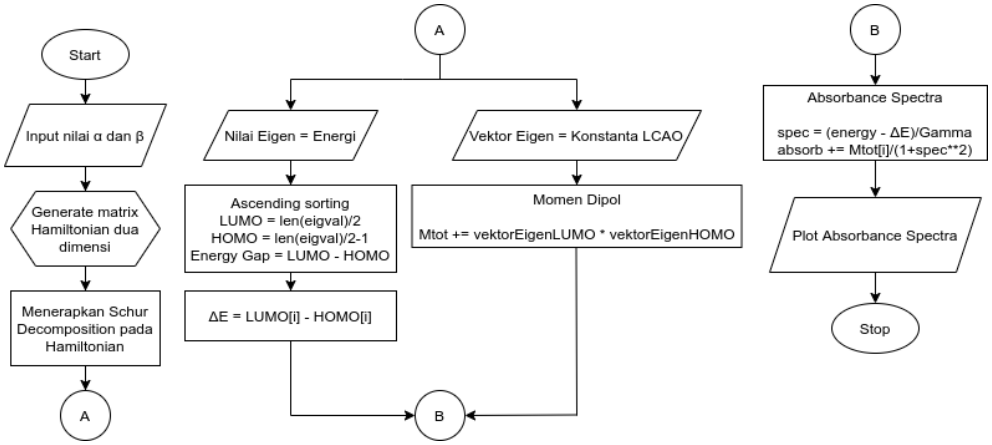
BAB III

METODE PENELITIAN

Penelitian Kalkulasi Nilai Eigen berbasis Matriks Hamiltonian menggunakan Teknik *Block Matrix* : Studi Kasus Pada *Graphene* dilakukan dengan 3 tahapan, yaitu perancangan algoritma penelitian, perancangan kode komputasi dan simulasi data. Perancangan algoritma penelitian akan membuat cara kerja aplikasi yang akan dibuat dimulai dari *Start* hingga komputasi berakhir. Dalam perancangan kode komputasi akan dibuat kode berbahasa *Python* yang akan dieksekusi oleh *Python Compiler* dan PySpark. Tahap terakhir adalah kalkulasi data data. Simulasi data akan mensimulasikan kode yang sudah dibuat menggnakan perangkat komputer dengan spesifikasi yang ditentukan. Dari hasil kalkulasi data ini akan dilihat hasil grafik celah energi dan spektrum absorbansi lalu performa komputasi yang dijelaskan dengan kecepatan komputasi yang dihasilkan dari variasi yang diterapkan.

3.1 Perancangan Algoritma Penelitian

Langkah-langkah penelitian dapat dilihat pada diagram alir penelitian pada Gambar 3.1



Gambar 3.1: Diagram Alir Penelitian

Dalam penelitian ini, akan dilakukan dengan beberapa tahap, yaitu pendefinisian, pemecahan matriks utama menjadi sub matriks kecil, diagonalisasi matriks, dan subtitusi untuk dapat keluaran besar energi orbital molekul dari graphene dan laju konvergensi yang didapatkannya. Diawali dengan penginputan besar α dan β dari Graphene lalu akan dibuat matriks Hamiltoniannya. Lalu, matriks Hamiltoniannya akan dioperasikan menggunakan dekomposisi Schur untuk dapat dikerjakan secara paralel. Keluaran nya berupa nilai eigen dan vektor eigen. Nilai eigen sebagai energi orbital, dan vektor eigen merupakan konstanta fungsi gelombangnya. Dari kedua keluaran ini, akan dicari besar absorpsi setiap perubahan energi eksitasinya lalu dibuatkan grafik.

3.2 Alat yang Digunakan

3.2.1 Perangkat Lunak

1. *Python* versi 3.8.5
2. *Apache Spark* versi 3.0.1 dengan *PySpark*
3. *Apache Hadoop* versi 3.14

4. *Visual Studio Code* versi 1.56.2
5. *Gnuplot* versi 5.4 *patchlevel* 1

3.2.2 Perangkat Keras

1. Server KST
   (a) Prosessor : Intel Xeon E3-1225; 4 Core @ 3.7 GHz
   (b) RAM : 8 GB
   (c) Sistem Operasi : Ubuntu 18.04 LTS
2. Server Google Cloud Instance Iowa (US)
   (a) Prosessor : Intel Xeon Skylake; 4 vCore/vCPU @ 3.7GHz
   (b) RAM : 16 GB
   (c) Sistem Operasi : Ubuntu 20.04 LTS

3.3 Perancangan Kode Komputasi

3.3.1 Perancangan Matriks Hamiltonian

Pada perancangan kode komputasi ini akan membuat kode dalam bahasa *Python* yang akan dikonversi menjadi kode yang dapat dieksekusi di *PySpark*. Matriks Hamiltonian yang dimiliki oleh Graphene bersifat matriks dua dimensi. Lalu dalam penentuan ikatan nya akan dilakukan menggunakan penentuan posisi atom acak dua dimensi. Penentuan posisi $\vec{A}$ yaitu dengan

$$\vec{A} = A_x + A_y \tag{3.1}$$

$$A_x = rand * L_x A_y = rand * L_y \tag{3.2}$$

dengan rand = angka acak, dan L_x = L_y = panjang horizontal dan vertikal.

Untuk menentukan indeks cell dari matriks dapat menggunakan rumus:

$$index = A_y * L_x + A_x \tag{3.3}$$

Maka, dari indeks yang dihasilkan akan menjadi matriks seperti berikut

$$\begin{bmatrix} index_{11} & index_{12} & index_{13} \\ index_{21} & index_{22} & index_{23} \\ index_{31} & index_{32} & index_{33} \end{bmatrix} \tag{3.4}$$

atau jika dijadikan angka indeks,

$$\begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} \tag{3.5}$$

angka indeks inilah yang akan digunakan dalam pembuatan fungsi matriks Hamiltonian.

Kemudian karena kita akan mencari koordinat ($index_x$, $index_y$), maka kita harus mencari A_x dan A_y nya terlebih dahulu.

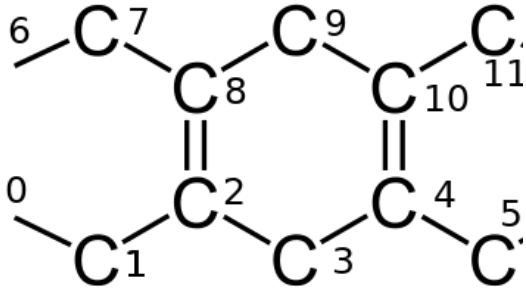
$$\begin{aligned} i_y &= index_x / L_x \\ i_x &= index_x - i_y * L_x \end{aligned} \tag{3.6}$$

Dari i_x dan i_y yang telah didapatkan, kemudian dicari $index_j$ nya dengan rumus,

$$index_j = j_y * L_x + j_x \tag{3.7}$$

dengan $j_y = iy + dy$ dan $j_x = ix + dx$.

Untuk menentukan atom mana saja yang terikat pada Graphene, kita bisa menurunkan dari ilustrasi pada Gambar 3.2.



Gambar 3.2: Indeks Struktur Empirik Graphene

Dari gambar terlihat bahwa tidak semua atom saling berikatan. Seperti contoh atom dengan indeks 1 tidak berikatan dengan atom indeks 7 dan hanya berikatan dengan atom indeks 0 dan 2. Lalu atom indeks 2 berikatan dengan indeks 1, 3 dan 8. Maka disini dilakukan pengondisian menggunakan vektor dengan rumus,

$$rad = \sqrt{dx^2 + dy^2} \tag{3.8}$$

Hasil yang didapatkan digunakan untuk pengondisian dalam variabelisasi elemen α dan β seperti pada persamaan (2.8) dengan pengondisian sebagai berikut:

$$\begin{cases} rad = 0,\ H[i][j] = \alpha \\ rad = 1,\ mod(L_x) = 0,\ abs(i-j) = L_x,\ danmod(i) = 1,\ H[i][j] = 0 \\ rad = 1,\ mod(L_x) = 1,\ mod(iy) = 0,\ abs(i-j) = L_x,\ danmod(i) = 1,\ H[i][j] = 0 \\ rad = 1,\ mod(L_x) = 1,\ mod(iy) = 1,\ abs(i-j) = L_x,\ danmod(i) = 0,\ H[i][j] = 0 \\ else,\ H[i][j] = \beta \end{cases} \tag{3.9}$$

3.3.2 Perancangan Kalkulasi Celah Energi

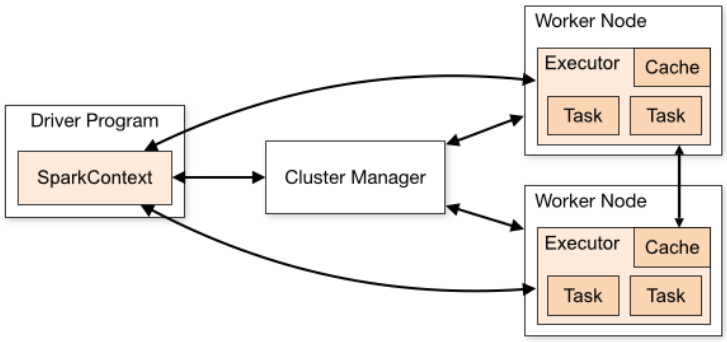
Matriks Hamiltonian yang didapat lalu diaplikasikan dengan teknik *Block Matrix* dengan mnenggunakan dekomposisi Schur. Dengan mengaplikasikan persamaan (2.15), akan didapatkan dua buah output yaitu nilai eigen dan vektor eigen. Nilai eigen akan menjadi energi orbital, dan vektor eigen akan menjadi koefisien fungsi gelombangnya (yaitu c pada persamaan (2.1)).

Energi orbital nya akan di urutkan secara *ascending* untuk menempatkan energi sesuai konsep HOMO dan LUMO. Setengah atas adalah HOMO, dan setengah bawah adalah LUMO. Untuk mencari celah pita nya, dapat menghitung perbedaan energi HOMO dan LUMO terluarnya,

$$E_{gap} = E_{LUMO\ terluar} - E_{HOMO\ terluar}\ (eV) \tag{3.10}$$

3.3.3 Perancangan Kalkulasi Celah Energi dengan *Python* dan *PySpark*

Untuk mencari celah energi nya, akan dilakukan menggunakan 2 buah aplikasi yaitu *Python* dan *PySpark*. Untuk Python akan dilakukan dengan 3 variasi yaitu *Serial Processing*, *Multi Processing*, dan *Threading*. Dari 4 buah variasi komputasi ini, akan dilihat performa komputasinya melalui jumlah waktu yang dibutuhkan untuk memproses matriks Hamiltonian. Khusus untuk PySpark, akan dijalankan menggunakan (spark−submit) dengan 4 core di setiap workernya. Skema kerja dari PySpark adalah sebagai berikut:



Gambar 3.3: Skema Kerja PySpark

Dalam Gambar 3.3 terlihat bahwa Spark menggunakan arsitektur master / slave. Spark memiliki satu koordinator pusat (*driver*) yang berkomunikasi dengan banyak *worker* (eksekutor) yang tersebar. *Driver* dan setiap *worker* menjalankan proses Python mereka masing-masing.

3.3.4 Perancangan Kalkulasi Absorbansi terhadap Perubahan Energi Eksitasi

Hasil dari kalkulasi energi eigen, lalu akan dihitung besar energi eksitasi dari HOMO ke LUMO dengan mengurangi tiap orbital LUMO dengan HOMO nya

$$\Delta E = E_{LUMO} - E_{HOMO}\ (eV) \tag{3.11}$$

Untuk vektor eigen nya, dioperasikan untuk mencari momen dipol yang dimiliki oleh Graphene dengan mengalikan tiap kolom di daerah LUMO dengan tiap kolom di daerah HOMO. Secara matematisnya,

$$M_{total} = \sum_{i}^{N_{atoms}} c_{[i,\ LUMO]} * c_{[i,\ HOMO]} \tag{3.12}$$

Dengan menggabungkan hasil dari ΔE pada persamaan (3.11) dengan M_{total} pada persamaan (3.12), maka kita dapat mencari nilai absorbansinya dengan menggunakan persamaan (2.10).

BAB IV

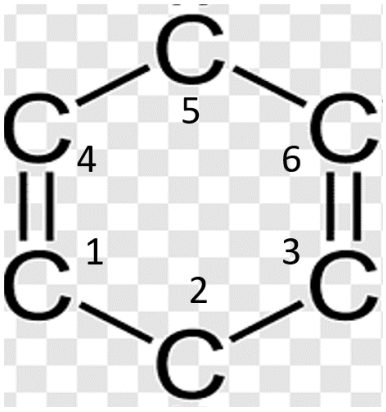
HASIL DAN PEMBAHASAN

4.1 Hasil Kode Komputasi

Pada bab ini, akan membahas mengenai hasil dari perancangan kode komputasi, seperti kalkulasi celah energi, kalkulasi celah energi dengan *Python* dan *PySpark* dan kalkulasi absorbansi terhadap perubahan energi eksitasi.

4.1.1 Hasil Matriks Hamiltonian

Dari perancangan matriks Hamiltonian, akan didapatkan sebuah matriks $m_x n$ 2 dimensi. Matriks ini akan menjelaskan keberadaan partikel dan ikatan yang dimiliki oleh partikel tersebut dengan tetangga terdekatnya. Karena hanya mengambil tetangga terdekatnya, maka jarak antar atomnya dapat dianggap satu ($r_{ij} = 1$). Untuk membuat matriks Hamiltoniannya, maka harus terlebih dahulu menentukan panjang atom di sumbu x dan panjang atom di sumbu y.



Gambar 4.1: Struktur Empirik Benzene untuk Hückel

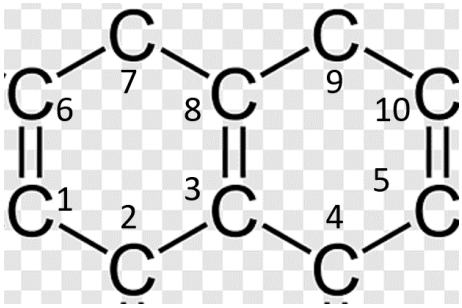
Untuk benzene, maka panjang atom x nya adalah 3, dan panjang atom y nya adalah 2. Matriks Hamiltonian yang didapatkan nya seperti berikut.

```
array([[-10.7 ,  -1.58,   0.  ,  -1.58,   0.  ,   0.  ],
       [ -1.58, -10.7 ,  -1.58,   0.  ,   0.  ,   0.  ],
       [  0.  ,  -1.58, -10.7 ,   0.  ,   0.  ,  -1.58],
       [ -1.58,   0.  ,   0.  , -10.7 ,  -1.58,   0.  ],
       [  0.  ,   0.  ,   0.  ,  -1.58, -10.7 ,  -1.58],
       [  0.  ,   0.  ,  -1.58,   0.  ,  -1.58, -10.7 ]])
```

Gambar 4.2: Matriks Hamiltonian Benzene

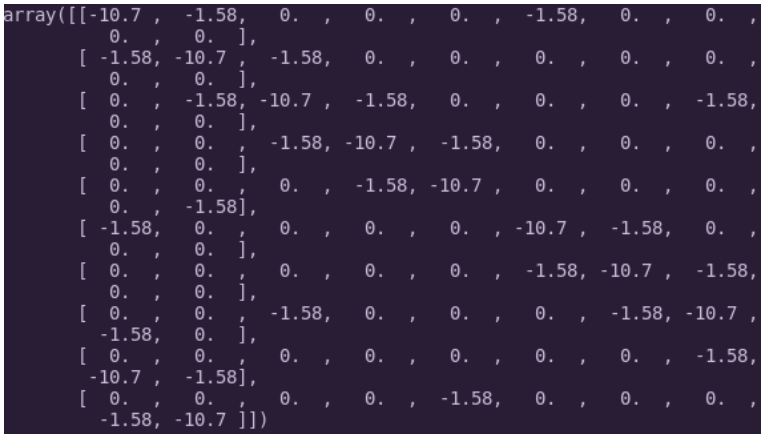
Jika kita validasi Gambar 4.2 dengan ikatan benzene pada Gambar 4.1, maka terlihat bahwa tidak semua atom akan berikatan satu sama lain. Jika mengambil contoh atom C_1, maka hanya akan berikatan dengan atom C_4 dan C_2. Maka atom C_1 akan bernilai α dan ikatan dengan atom C_4 dan C_2 bernilai β. Maka matriks dengan indeks H_{11} bernilai $-10,7$ eV, lalu H_{12} dan H_{14} bernilai $-1,58$ eV, begitupun untuk indeks lainnya.

Untuk atom karbon, jika diperbanyak, maka indeks ikatannya pun akan berbeda. Misalkan untuk senyawa C_{10}, struktur yang dimilikinya akan seperti berikut,



Gambar 4.3: Struktur Empirik C_{10} untuk Hückel

dan matriks hamiltonian yang didapatkannya adalah



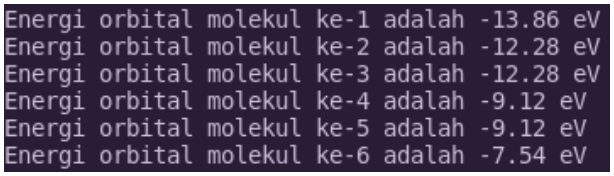
Gambar 4.4: Matriks Hamiltonian C_{10}

Terlihat antara struktur C_{10} (Gambar 4.3) dengan matriks hamiltoniannya (Gambar 4.4) memiliki indeks yang sama. Sebagai contoh, untuk atom C_3 akan berikatan dengan atom C_8, C_2 dan C_4 yang menyebabkan terdapat nilai β pada matriks H_{38}. H_{34} dan H_{32}. Sedangkan untuk atom C_2 hanya berikatan dengan atom C_1 dan C_3 saja yang menyebabkan nilai β hanya terdapat pada matriks H_{21} dan H_{23}.

Karena hanya melibatkan ikatan dengan tetangga terdekat (*First-Nearest Neighbor*) sesuai asumsi Hückel, maka nilai β (sebesar -1.58 eV) hanya untuk ikatan terdekatnya, sedangkan interaksi dengan atom lainnya diabaikan.

4.1.2 Hasil Kalkulasi Celah Energi

Dari matriks Hamiltonian yang didapatkan, kemudian dihitung nilai eigen nya. Nilai eigen ini yang menjadi energi yang dimiliki tiap orbital molekul. Nilai energi orbital molekul dari benzene adalah,

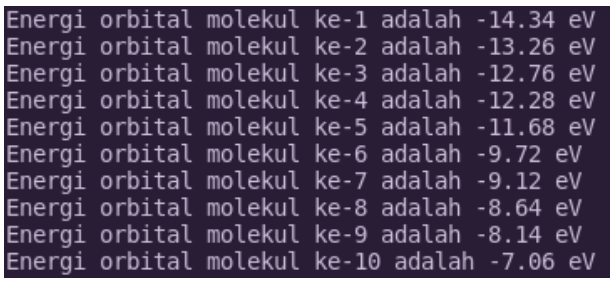


Gambar 4.5: Energi Orbital Molekul Benzene

dengan mengurangi energi HOMO terluar dengan LUMO terluar, maka didapatkan nilai celah energinya.

$$E_{gap}(benzene) = (-12,28 - (-9,12))eV = -3,16\ eV \tag{4.1}$$

Apabila jumlah atom karbon ditambah menjadi senyawa C_{10}, maka energi orbital molekul nya menjadi 10 keadaan dengan energi tiap keadaannya sebesar,

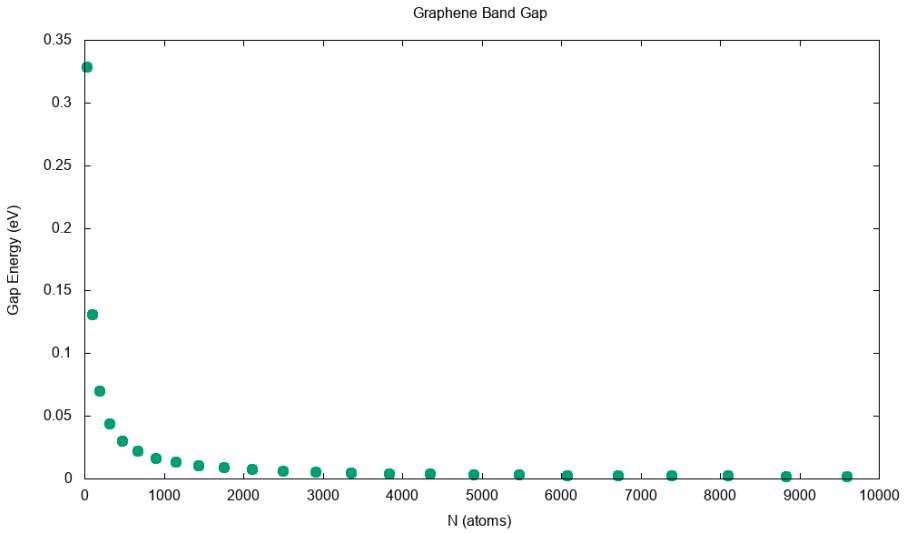


Gambar 4.6: Energi Orbital Molekul C_{10}

Terlihat bahwa jika jumlah atom karbon diperbanyak, perbedaan energi tiap orbitalnya akan mengecil. Ini menyebabkan celah energinya pun akan semakin mengecil. Saat di plot kedalam grafik, maka akan terlihat jelas penurunan celah energinya seperti pada Gambar 4.7 dan pada Tabel 4.3

Tabel 4.3: Tabel Celah Energi terhadap Banyaknya Atom Karbon

N (atom)	Celah Energi (eV)
6	3,1599
22	0,6942
36	0,3284
100	0,1306
196	0,0698
324	0,0434
484	0,0296
676	0,0214
900	0,0163
...	...
8100	0.0019
8836	0,0017
9604	0,0016



Gambar 4.7: Grafik Celah Energi terhadap Banyaknya Atom

4.1.3 Hasil Kalkulasi Celah Energi dengan *Python* dan *PySpark*

Karena ukuran matriks yang sangat besar, maka dilakukanlah beberapa teknik untuk mencari pemrosesan komputasi dengan waktu tercepat. Untuk itu, dilakukanlah komputasi dengan Python (menggunakan *Serial Processing*,

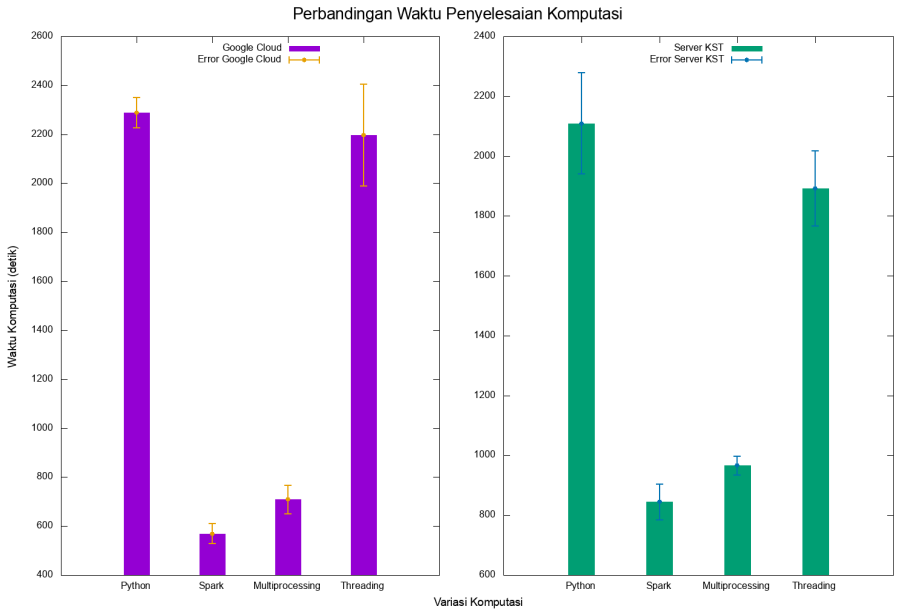
Multiprocessing dan *Threading*) dan PySpark. Dengan menggunakan matriks berukuran $36 * 36$ hingga $4900 * 4900$, didapatkan data waktu pemrosesan untuk Server Google Cloud dan Server KST sebagai berikut,

Tabel 4.4: Tabel Performa Komputasi dengan Server Google Cloud

Variasi Komputasi	Eksekusi ke-1	Eksekusi ke-2	Eksekusi ke-3	Eksekusi ke-4	Eksekusi ke-5	Rata-rata	Stddev
Python	2205,32	2377,87	2266,75	2254,75	2337,57	2288,46	61,53
Spark	528,73	537,24	544,09	612,98	622,35	569,08	40.08
Multiprocessing	615,76	786,04	738,32	677,24	726,87	708,84	57,99
Threading	2007,08	2035,64	2416,64	2040,86	2480,25	2196,46	207,34

Tabel 4.5: Tabel Performa Komputasi dengan Server KST

Variasi Komputasi	Eksekusi ke-1	Eksekusi ke-2	Eksekusi ke-3	Eksekusi ke-4	Eksekusi ke-5	Rata-rata	Stddev
Python	1919,96	2221,53	1890,03	2300,64	2214,35	2109,31	169,81
Spark	819,59	762,35	866,24	943,08	832,33	844,72	59,51
Multiprocessing	915,15	986,36	964,46	1008,46	954,28	965,74	31,44
Threading	1758,13	1781,08	2095,36	1969,45	1859,09	1892,62	125,43



Gambar 4.8: Grafik Performa Komputasi

Dari Gambar 4.8 terlihat bahwa menggunakan PySpark memiliki keunggulan dalam waktu komputasi. Waktu yang dibutuhkan untuk menghitung matriks berukuran $36 * 36$ hingga $4900 * 4900$ hanya sebesar $569,08$ detik untuk Server

Google Cloud dan $844,72$ detik untuk Server KST. Pada urutan kedua tercepat yaitu menggunakan multiprocessing. Untuk multiprocessing dan PySpark memiliki cara kerja yang sama, yaitu menggunakan paralelisasi *core* yang dimiliki oleh prosessor. Namun, untuk PySpark menggunakan paralelisasi kedalam jumlah pekerja (*worker*) yang akan mengerjakan suatu tugas (*task*) dengan *In-Memory Processing*, yaitu proses yang akan dilakukan melibatkan *memory*. Untuk *load* dari server dapat dilihat di Lampiran 6 dan Lampiran 7. Kemudian jika dibandingkan antara penggunaan Server Google Cloud dengan Server KST, terdapat keunggulan untuk pemrosesan Python dan Threading untuk Server KST dan keunggulan untuk pemrosesan PySpark dan Multiprocessing untuk Server Google Cloud.

4.1.4 Hasil Kalkulasi Absorbansi terhadap Perubahan Energi Eksitasi

Hasil dari energi orbital kemudian dicari besar eksitasi dari HOMO ke LUMO tiap keadaannya. Untuk benzene, maka energi eksitasi tiap keadaan HOMO ke LUMO nya adalah,

Tabel 4.6: Tabel Energi Eksitasi tiap Keadaan HOMO ke LUMO

Eksitasi Orbital	ΔE (eV)
4 ke 3	3,16
4 ke 2	3,16
4 ke 1	4,74
5 ke 3	3,16
5 ke 2	3,16
5 ke 1	4,74
6 ke 3	4,74
6 ke 2	4,74
6 ke 1	6,32

Kemudian dari hasil vektor eigen, di masukkan kedalam persamaan 2.1 untuk membuat persamaan *linear combination atomic orbital* dan menjadi,

$$\begin{cases} \psi_1 = -0,408\phi_1 - 0,577\phi_2 + 0,408\phi_3 - 0,577\phi_4 - 0,031\phi_5 + 0,045\phi_6 \\ \psi_2 = -0,408\phi_1 - 0,289\phi_2 - 0,408\phi_3 + 0,289\phi_4 - 0,515\phi_5 - 0,520\phi_6 \\ \psi_3 = -0,408\phi_1 + 0,289\phi_2 + 0,408\phi_3 + 0,289\phi_4 - 0,484\phi_5 + 0,477\phi_6 \\ \psi_4 = -0,408\phi_1 - 0,289\phi_2 - 0,408\phi_3 + 0,289\phi_4 + 0,484\phi_5 + 0,477\phi_6 \\ \psi_5 = -0,408\phi_1 + 0,289\phi_2 + 0,408\phi_3 + 0,289\phi_4 + 0,515\phi_5 - 0,521\phi_6 \\ \psi_6 = -0,408\phi_1 + 0,577\phi_2 - 0,408\phi_3 - 0,577\phi_4 + 0,031\phi_5 + 0,045\phi_6 \end{cases} \tag{4.2}$$

Konstanta fungsi gelombang inilah yang akan menginterpretasikan momen dipol yang dimiliki oleh Benzene. Untuk mencari momen dipol, dapat menggunakan rumus 3.12. Hasil yang didapatkannya adalah,

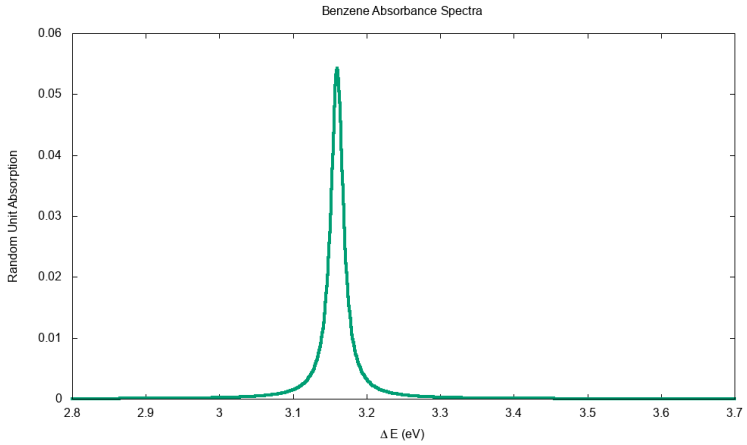
Tabel 4.7: Tabel Momen Dipol Transisi

Transisi Orbital	Momen Dipol Transisi (eV)
4 ke 3	0,054
4 ke 2	1,249*10^{-15}
4 ke 1	7,772*10^{-16}
5 ke 3	1,804*10^{-16}
5 ke 2	1,284*10^{-16}
5 ke 1	-8,327*10^{-17}
6 ke 3	-2,689*10^{-16}
6 ke 2	-4,996*10^{-16}
6 ke 1	-1.183*10^{-15}

Momen dipol transisi terbesar adalah saat transisi orbital 4 ke 3. Hal ini

disebabkan karena transisi tersebut terletak pada titik terluar yang menyebabkan transisinya akan membutuhkan energi yang besar, dan transisi orbital 4 ke 3 merupakan transisi yang terjadi pada celah energi dari senyawa tersebut.

Dari hasil perubahan energi eksitasi tiap orbital (Tabel 4.6) dan hasil momen dipol transisi (Tabel 4.7) akan dihitung besar absorbansi terhadap perubahan energi eksitasi tiap orbital menggunakan persamaan 2.10. Jika diplot, grafik absorpsi terhadap ΔE nya menjadi,

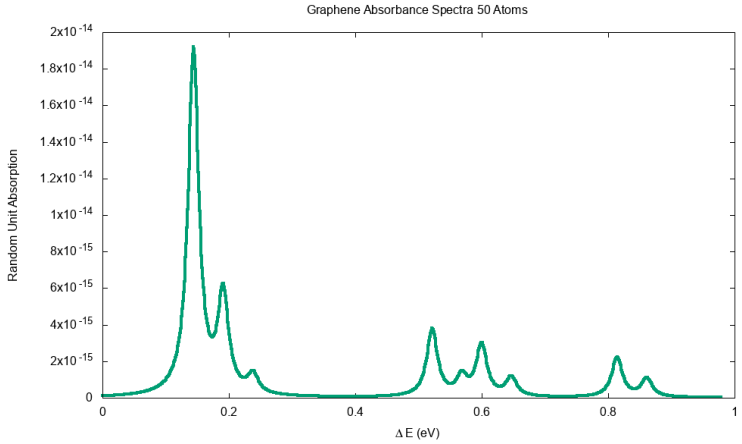


Gambar 4.9: Spektrum Absorbansi terhadap Perubahan Energi Eksitasi pada Benzene

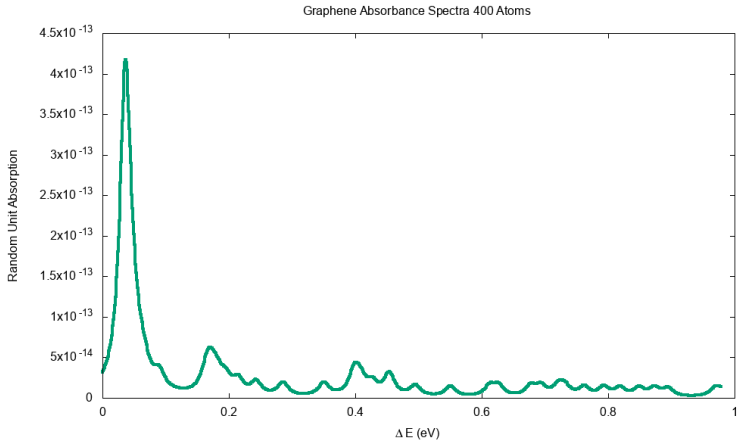
Dari gambar 4.9, terlihat bahwa titik puncak penyerapan energi terjadi pada ΔE sebesar $\pm$ 3.15 eV dan ini adalah besar celah energi yang dimiliki oleh Benzene. Hal ini disebabkan karena titik absorbsi energi menggambarkan besar energi yang akan diserap oleh elektron, dan energi yang diserap akan membuat elektron tereksitasi ke keadaan dengan energi yang lebih besar.

Apabila jumlah atom karbon diperbanyak, maka puncak absorbansi yang terjadi akan bergeser ke kiri (mendekati ΔE = 0). Terlihat pada Gambar 4.10 dan Gambar

4.11.



Gambar 4.10: Spektrum Absorbansi terhadap Perubahan Energi Eksitasi pada Graphene 50 Atom



Gambar 4.11: Spektrum Absorbansi terhadap Perubahan Energi Eksitasi pada Graphene 400 Atom

Selain itu, terdapat puncak-puncak kecil yang menunjukkan masih ada energi yang terserap oleh elektron. Ini menunjukkan bahwa Graphene bersifat metalik yaitu sebagai konduktor yang baik akibat perpindahan atau eksitasi elektron yang terjadi sangatlah mudah.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Telah dirancang algoritma untuk membangun matriks Hamiltonian dan kalkulasi spektrum absorbsi menggunakan bahasa pemrograman Python. Data-data yang akan diamati meliputi: celah energi (*gap energy*), energi eksitasi tiap tingkat orbital molekul (ΔE), momen dipol transisi, dan absorbsi. Celah energi yang didapatkan akan mengecil jika semakin banyak atom karbon yang dimiliki oleh Graphene. Ini disebabkan karena saat atom karbon bertambah, jarak energi antar orbital molekulnya akan semakin kecil dan menyebabkan elektron lebih mudah untuk bereksitasi. Dalam penghitungan besar energi orbital molekul, dilakukan dengan menggunakan variasi komputasi Python, *PySpark*, *Multiprocessing* dan *Threading*. Dari keempat variasi tersebut, PySpark memiliki keunggulan dalam waktu komputasi yang dibutuhkan dengan rata-rata $569,08$ detik untuk Server Google Cloud dan $844,72$ detik untuk Server KST. Setelah menghitung celah energi, lalu dicari energi eksitasi tiap tingkat orbital dan momen dipol transisinya dan akan didapatkan besar absorbansi tiap perubahan energi eksitasi. Jika di plot kedalam grafik, akan terlihat puncak absorbsi yang akan menginterpretasikan besar celah energi yang dimiliki oleh Graphene. Hal ini karena momen dipol transisi akan menggambarkan besar energi transisi orbital

molekulnya, dan energi transisi tertinggi akan menggambarkan transisi dari keadaan HOMO ke LUMO yang merupakan celah energi. Puncak absorbansi akan semakin menggeser mendekati $\Delta E = 0$ saat atom karbon nya bertambah dan juga terdapat puncak-puncak kecil dari absorbansi yang menunjukkan Graphene merupakan bahan dengan sifat metalik dan konduktor yang baik.

5.2 Saran

Dari penelitian yang sudah dilakukan, terdapat beberapa masukan untuk penelitian selanjutnya yaitu:

1. Membuat plot spektrum absorbansi terhadap panjang gelombang dalam nanometer untuk melihat besar penyerapan energi di setiap panjang gelombangnya sesuai kerja UV-Vis Spektrometer

2. Membuat kode untuk mengkalkulasi spektrum absorbsi Graphene Oxide dan Reduced Graphene Oxide

3. Menganimasikan perilaku orbital molekul pada Graphene untuk mempermudah pembaca dalam memahami peristiwa tersebut

4. Melihat pengaruh *Meta-Stable Energy* yang menyebabkan terdapat energi yang tersisa pada grafik absorbsi

5. Menggunakan performa Apache Spark dengan *full-power* agar performa komputasi yang dihasilkan lebih baik

DAFTAR PUSTAKA

[1] J. C. Slater. *Introduction to Chemical Physics*. 1939.

[2] ACS. *Organic Chemistry*. 2020. URL: https://www.acs.org/content/acs/en/careers/college (visited on 07/27/2020).

[3] Zhen Zhen and Hongwei Zhu. *Structure and properties of graphene*. Elsevier Inc., 2017, pp. 1–12. ISBN: 9780128126516. DOI: 10.1016/B978-0-12-812651-6.00001-X. URL: http://dx.doi.org/10.1016/B978-0-12-812651-6.00001-X.

[4] B. Ribeiro and W Kusuma. "Aplikasi Metode Huckel dan Slater pada Orbital Molekul Untuk Menentuan Energi Ikatan Molekul". In: (), pp. 1–28.

[5] P. Ni. "Band gap of graphene nanoribbons calculated using Huckel molecular orbital theory". In: *International Journal of Engineering and Applied Sciences* 2.3 (2015), p. 257979. ISSN: 2394-3661.

[6] J. Clayden, N. Greeves, and S. Warren. *Organic Chemistry*. 2nd. Oxford, 2012.

[7] C. B. Clemons et al. "Continuum plate theory and atomistic modeling to find the flexural rigidity of a graphene sheet interacting with a substrate". In: *Journal of Nanotechnology* 2010 (2010). ISSN: 16879503. DOI: 10.1155/2010/868492.

[8] Ben. *Metana*. 2016. URL: https://id.wikipedia.org/wiki/Berkas:Methane-2D-dimensions.svg.

[9] Sivabrata Sahu and G C Rout. “Band gap opening in graphene: a short theoretical study”. In: *International Nano Letters* 7.2 (2017), pp. 81–89. ISSN: 2008-9295. DOI: 10.1007/s40089-017-0203-5.

[10] Michael Berger. *What is Graphene?* 2019. URL: https://www.nanowerk.com/what_is_graphene.php.

[11] Verjari. *Graphene clothing: Exceptional sports properties*. URL: https://www.verjari.fr/en/blogs/articles/vetements-graphene-sport-clothes.

[12] Manijeh Razeghi. “The carbon atom”. In: *The Mystery of Carbon*. 2053-2563. IOP Publishing, 2019, pp. 1–12. ISBN: 978-0-7503-1182-3. DOI: 10.1088/2053-2563/ab35d1ch1. URL: http://dx.doi.org/10.1088/2053-2563/ab35d1ch1.

[13] Monisha Chakraborty and M. Saleem J. Hashmi. *Graphene as a Material – An Overview of Its Properties and Characteristics and Development Potential for Practical Applications*. Elsevier Ltd., 2018, pp. 1–17. ISBN: 9780128035818. DOI: 10.1016/b978-0-12-803581-8.10319-4. URL: http://dx.doi.org/10.1016/B978-0-12-803581-8.10319-4.

[14] Satiye Korkmaz and İ Af. “Graphene and graphene oxide based aerogels : Synthesis , characteristics and supercapacitor applications”. In: 27.September 2019 (2020). DOI: 10.1016/j.est.2019.101038.

[15] F. Albert Coton. *Chemical applications of group theory*. 3rd ed. New York: Wiley, 1990, p. 102.

[16] Rustam E Siregar. *MEKANIKA KUANTUM MOLEKUL Struktur Elektronik Atom dan Molekul*. Jatinangor: Unpad Press, 2014. ISBN: 9786029238624.

[17] Aradhana Anil and S. Ramasesha. *Huckel theory and Band theory: Computational approach*. Tech. rep. URL: http://reports.ias.ac.in/report/14114/huckel-theory-and-band-theory-computational-approach.

[18] Rustaman. "KEADAAN-KEADAAN TRANSISI MOLEKUL". In: *Karya Tulis Ilmiah Kimia Unpad* (2008), pp. 1–21.

[19] M J Mączyński. "A SIMPLE DERIVATION OF HUCKEL'S MOLECULAR ORBITALS IN THE FRAMEWORK OF QUANTUM LOGIC". In: *Reports on Mathematical Physics* 30.1 (1991), pp. 81–88. DOI: https://doi.org/10.1016/0034-4877(91)90042-L.

[20] Yutaka Imamura et al. "Extrapolation of polymer gap by combining cluster and periodic boundary condition calculations with Hückel theory". In: *Chemical Physics Letters* 707 (2018), pp. 44–48. ISSN: 00092614. DOI: 10.1016/j.cplett.2018.07.023. URL: https://doi.org/10.1016/j.cplett.2018.07.023.

[21] Risdiana. *Diktat Pengantar Fisika Zat Padat*. Jatinangor: Unpad, 2014.

[22] Tomgelly. *Molecule HOMO-LUMO Diagram*. 2010. URL: https://commons.wikimedia.org/wiki/File:Molecule_HOMO-LUMO_diagram.svg.

[23] Sarah Faizi and Craig Fisher. *Pauli Exclusion Principle*. 2021. URL: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Electronic_Structure_of_Atoms_and_Molecules/Electronic_Configurations/Pauli_Exclusion_Principle (visited on 07/14/2021).

[24] J P Bird. "Semiconductors: An Introduction". In: ed. by K H Jürgen Buschow et al. Oxford: Elsevier, 2002, pp. 1–16. ISBN: 978-0-08-043152-9. DOI: https://doi.org/10.1016/B0-08-043152-6/01854-4. URL: https://www.sciencedirect.com/science/article/pii/B0080431526018544.

[25] Wildan Abdussalam. "Dynamics of Rydberg atom lattices in the presence of noise and dissipation". PhD thesis. Technischen Universität Dresden, 2017.

[26] Tim Davis. "Block matrix methods: Taking advantage of high-performance computers". In: (2004).

[27] William Ford. *Numerical Linear Algebra with Applications: Editorial*. Vol. 10. 7. USA: Elsevier, 2003, p. 561. ISBN: 9780123944351. DOI: 10.1002/nla.335.

[28] Patrick Pisciuneri. *How Target Performance Tunes Machine Learning Applications*. 2016.

[29] Muhammad Saqlain et al. *A Classification Model for Predicting Heart Failure in Cardiac Patients*. 2017, pp. 36–43. ISBN: 978-3-319-51233-4. DOI: 10.1007/978-3-319-51234-1_6.

[30] Oracle. *Big Data Defined*. 2020. URL: https://www.oracle.com/big-data/what-is-big-data/.

[31] Nataliya Shakhovska et al. "Big data processing technologies in distributed information systems". In: *Procedia Computer Science* 160.2018 (2019), pp. 561–566. ISSN: 18770509. DOI: 10.1016/j.procs.2019.11.047. URL: https://doi.org/10.1016/j.procs.2019.11.047.

[32] Jose Moura and Carlos Serrao. "Security and Privacy Issues of Big Data". In: 2015. ISBN: 9781466685055. DOI: 10.4018/978-1-4666-8505-5.ch002.

[33] Seth Schaafsma. *Big Data: The 6 Vs You Need to Look at for Important Insights*. 2020. URL: https://www.motivaction.nl/en/news/blog/big-data-the-6-vs-you-need-to-look-at-for-important-insights.

[34] Nurfauzi Fadillah. "Rancang Bangun Prototipe Sistem Pemrosesan Big Data untuk Analisis dan Pemantauan Sistem Daya Listrik". PhD thesis. Universitas Padjadjaran, 2020.

[35] Matthew Stewart. *Handling Big Datasets for Machine Learning - Towards Data Science*. 2019. URL: https://towardsdatascience.com/machine-learning-with-big-data-86bcb39f2f0b.

[36] Chynthia Harvey. *Big Data Pros and Cons*. 2018. URL: https://www.datamation.com/big-data/big-data-pros-and-cons.html.

[37] Apache. “Apache Spark”. In: (2018).

[38] Swatee Chand. *PySpark Programming – Integrating Speed With Simplicity*. 2020. URL: https://www.edureka.co/blog/pyspark-programming/.

[39] Krishna Kumar. *GitHub: Data Analytics*. 2018. URL: https://kks32-courses.gitbook.io/data-analytics/spark/rdd#partitions.

[40] Databricks. *Apache Spark*.

[41] Alex Bekker. *Spark vs. Hadoop MapReduce: Which big data framework to choose*. 2017. URL: https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce.

[42] Tutorialspoint. *PySpark - SparkContext*. URL: https://www.tutorialspoint.com/pyspark/pyspark_sparkcontext.htm.

DAFTAR LAMPIRAN

Lampiran 1: Kode Kalkulasi Huckel dan Absorbansi

```
1 import numpy as np
2 from scipy import linalg as la
3
4 def huckel(Lx, Ly):
5     """
6     Build hamiltonian matrix
7     """
8     N = Lx*Ly
9     H = np.zeros(shape = (N, N))
10    for atom_i in range(0,N):
11        alpha = -10.7
12        beta = -1.58
13        iy = int(atom_i/Lx)
14        ix = int(atom_i - Lx*iy)
15        #determine interaction with nearest neighbours
16        for dy in range(-1, 2):
17            for dx in range(-1, 2):
18                #determine coordinate atom_j
19                jx = ix + dx
20                jy = iy + dy
```

```
21
22                  if jx>=0 and jx<Lx and jy>=0 and jy<Ly:
23                      atom_j = int(jy*Lx + jx)
24                      rsquare = dx * dx + dy * dy
25                      rad = np.sqrt(rsquare)
26
27                      if rad == 0:
28                          H[atom_i][atom_j]= alpha
29                      elif rad == 1:
30                          if Lx%2 == 0 and abs(atom_i-atom_j)==Lx and atom_i%2 ==
                                1: #jika panjang x genap, dan atomi-atomj = panjang x
                                dan atom i ganjil
31                              H[atom_i][atom_j]=0
32                          elif Lx%2 == 1 and iy%2 == 0 and abs(atom_i-atom_j)==Lx
                                and atom_i%2==1: #jika panjang x ganjil, iy genap,
                                absolut dari atomi-atomj = panjang x dan atom i ganjil
33                              H[atom_i][atom_j]=0
34                          elif Lx%2 == 1 and iy%2 == 1 and abs(atom_i-atom_j)==Lx
                                and atom_i%2==0: #jika panjang x ganjil, iy ganjil, atomi
                                -atomj = panjang x dan atom i genap
35                              H[atom_i][atom_j]=0
36                          else :
37                              H[atom_i][atom_j] = beta
```

```
38                          # if abs(atom_i-atom_j)==Lx and atom_i%2==1:
39                              # H[atom_i][atom_j]=0
40                          # else:
41                              # H[atom_i][atom_j] = -1.58
42                      else:
43                          H[atom_i][atom_j] = 0
44
45          """
46          Calculating eigen energy with Schur Decomposition
47          """
48          # eigval_schur, Z = la.schur(H)
49          # eigval_schur = sorted(np.diag(eigval_schur))
50
51          """
52          Calculating eigen energy and eigen number
53          """
54          eigval, eignum = la.eig(H)
55          eigval = sorted(eigval) #sorting secara ascending
56
57          """
58          Calculating gap from eigen energy
59          """
60          indexHomo = int(len(eigval)/2-1) #nilai terkecil
```

```
61      indexLumo = int(len(eigval)/2) #nilai terbesar
62      Egap = eigval[indexLumo] - eigval[indexHomo] #dalam satuan elektronvolt
63
64      """
65      Calculating deltaE
66      """
67      dE = []
68      for indexEnergyLumo in range(0, indexLumo):
69          for indexEnergyHomo in range (0,indexHomo+1):
70              dE.append(eigval[indexEnergyLumo + indexLumo] - eigval[
                    indexHomo - indexEnergyHomo])
71
72      """
73      Calculating dipole moment
74      """
75      energyNumber = int(len(eignum))
76      M = []
77      for indexEnergyLumo in range(0, indexLumo):
78          for indexEnergyHomo in range (0, indexHomo+1):
79              mtot = 0
80              for indexC in range(0, energyNumber):
81                  mtot += eignum[indexC][indexEnergyLumo + indexLumo] *
                        eignum[indexC][indexHomo - indexEnergyHomo]
```

```
82                      M.append(mtot)
83              Ms = sorted(M, reverse=True)
84
85              """
86              Calculating Absorbance Spectra
87              """
88              def dipole_moment(Ndiff, Gamma, E):
89                  result = 0.0
90                  for atom_i in range (0, Ndiff):
91                      disc = E - dE[atom_i]
92                      disc /= Gamma # gamma = width nya
93                      result+=(Ms[atom_i]/(1+disc**2))
94                  return result
95              Npoints = 1000
96              x = np.linspace (0, 1, Npoints)
97              y_list = []
98              for i in range(0,Npoints):
99                  y = dipole_moment(N, 0.01, x[i])
100                 y_list.append(y.real)
101
102         return H, eigval, eignum, Egap, dE, Ms, x, y_list
103
104 def huckel_2(L):
```

```
105     """
106     Build hamiltonian matrix
107     """
108     N = L*2
109     H = np.zeros(shape = (N, N))
110     for atom_i in range(0,N):
111         alpha = -10.7
112         beta = -1.58
113         iy = int(atom_i/L)
114         ix = int(atom_i - L*iy)
115         #determine interaction with nearest neighbours
116         for dy in range(-1, 2):
117             for dx in range(-1, 2):
118                 #determine coordinate atom_j
119                 jx = ix + dx
120                 jy = iy + dy
121
122                 if jx>=0 and jx<L and jy>=0 and jy<2:
123                     atom_j = int(jy*L + jx)
124                     rsquare = dx * dx + dy * dy
125                     rad = np.sqrt(rsquare)
126
127                     if rad == 0:
```

```
128                 H[atom_i][atom_j]= alpha
129             elif rad == 1:
130                 if L%2 == 0 and abs(atom_i−atom_j)==L and atom_i%2 == 1:
                        #jika panjang x genap, dan atomi−atomj = panjang x dan
                        atom i ganjil
131                     H[atom_i][atom_j]=0
132                 elif L%2 == 1 and iy%2 == 0 and abs(atom_i−atom_j)==L
                        and atom_i%2==1: #jika panjang x ganjil, iy genap,
                        absolut dari atomi−atomj = panjang x dan atom i ganjil
133                     H[atom_i][atom_j]=0
134                 elif L%2 == 1 and iy%2 == 1 and abs(atom_i−atom_j)==L
                        and atom_i%2==0: #jika panjang x ganjil, iy ganjil, atomi
                        −atomj = panjang x dan atom i genap
135                     H[atom_i][atom_j]=0
136                 else :
137                     H[atom_i][atom_j] = beta
138                 # if abs(atom_i−atom_j)==L and atom_i%2==1:
139                     # H[atom_i][atom_j]=0
140                 # else:
141                     # H[atom_i][atom_j] = −1.58
142             else:
143                 H[atom_i][atom_j] = 0
144
```

```
145         """
146         Calculating eigen energy and eigen number
147         """
148         eigval, eignum = la.eig(H)
149         eigval = sorted(eigval)
150
151         """
152         Calculating gap from eigen energy
153         """
154         indexHomo = int(len(eigval)/2-1)
155         indexLumo = int(len(eigval)/2)
156         Egap = eigval[indexLumo] - eigval[indexHomo]
157
158         """
159         Calculating deltaE
160         """
161         dE = []
162         for indexEnergyLumo in range(0, indexLumo):
163             for indexEnergyHomo in range (0,indexHomo+1):
164                 dE.append(eigval[indexEnergyLumo + indexLumo] - eigval[
                        indexHomo - indexEnergyHomo])
165
166         """
```

```
167         Calculating dipole moment
168         """
169         energyNumber = int(len(eignum))
170         M = []
171         for indexEnergyLumo in range(0, indexLumo):
172             for indexEnergyHomo in range (0, indexHomo+1):
173                 mtot = 0
174                 for indexC in range(0, energyNumber):
175                     mtot += eignum[indexC][indexEnergyLumo + indexLumo] *
                            eignum[indexC][indexHomo - indexEnergyHomo]
176                 M.append(mtot)
177         Ms = sorted(M, reverse=True)
178
179         """
180         Calculating Absorbance Spectra
181         """
182         l = []
183         A = []
184         for i in range(len(Ms)):
185             for m in range(1,1000):
186                 el = 0.1 + m*0.0012
187                 l.append(el)
188                 a = 0
```

```
189                 a+=abs(Ms[i]**2/(dE[i]-(1.2/el-0.1*i)))
190                 A.append(a)
191
192     return H, eigval, eignum, Egap, dE, Ms, l, A
```

Lampiran 2: Kode Eksekusi dengan Python

```
1 from huckel import huckel
2 import numpy as np
3 from pprint import pprint
4
5 def main():
6     list_egap = []
7     N = np.arange(6,101,2)
8     for X in N:
9         H, eigval, eignum, Egap, dE, Ms, x, y_list = huckel(X,X)
10        list_egap.append(Egap)
11
12    file = open("spectra-graphene.txt", "w")
13    zipped = np.vstack((x,y_list)).T
14    np.savetxt(file, zipped, delimiter=' ')
15
16 if __name__ == "__main__":
17    main()
```

Lampiran 3: Kode Eksekusi dengan PySpark

```
1 from __future__ import print_function
2 from pyspark.sql import SparkSession
3 from pyspark import SparkContext, SparkConf
4 import numpy as np
5 import time
6 from operator import add
7 from huckel import huckel_2
8
9 #spark = SparkSession.builder.appName('Tugas Akhir').getOrCreate()
10 #sc = spark.sparkContext
11
12 conf = SparkConf().setAppName('Tugas Akhir').setMaster("spark://sparks.us-east1-c.c
       .sparkproject-319117.internal:7077")
13 # conf = SparkConf().setAppName('Tugas Akhir')
14 sc = SparkContext(conf=conf)
15 # spark = SparkSession.builder.appName('Tugas Akhir').getOrCreate()
16 # sc = spark.sparkContext
17
18 def main():
19     t0 = time.perf_conter()
20     x = np.arange(6,100,2)
21     sc.parallelize(x).map(huckel_2).reduce(add)
22     tn = time.perf_counter()
```

```
23      print(f'Waktu komputasi = {tn-t0} s')
24      sc.stop()
25
26 if __name__=='__main__':
27      main()
```

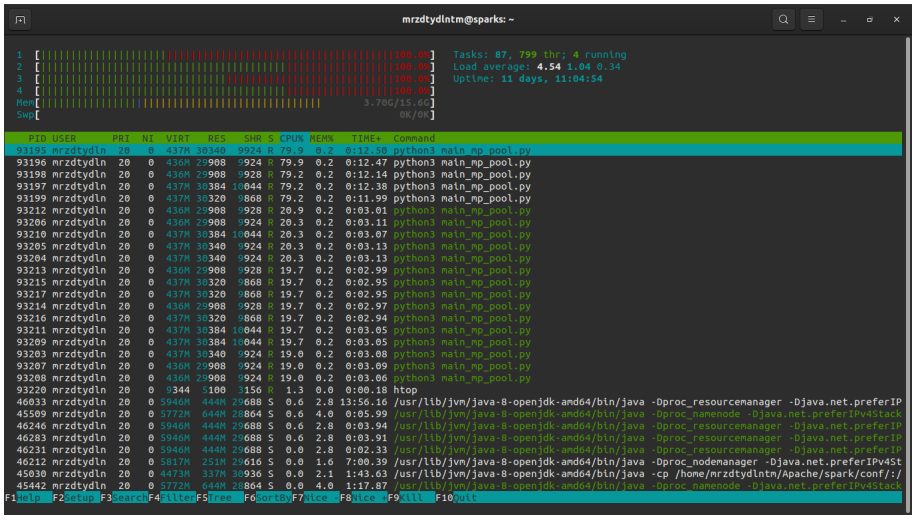
Lampiran 4: Kode Eksekusi dengan Multiprocessing

```
1 from huckel import huckel_2
2 import time
3 import multiprocessing as mp
4 import numpy as np
5
6 def main():
7      start = time.time()
8      X = np.arange(6,101,2)
9      with mp.Pool(5) as p:
10          res = p.map(huckel_2,X)
11          print(res[0][3])
12     stop = time.time()
13     print('Waktu komputasi = {} s'.format(stop-start))
14
15 if __name__ == "__main__":
16     main()
```

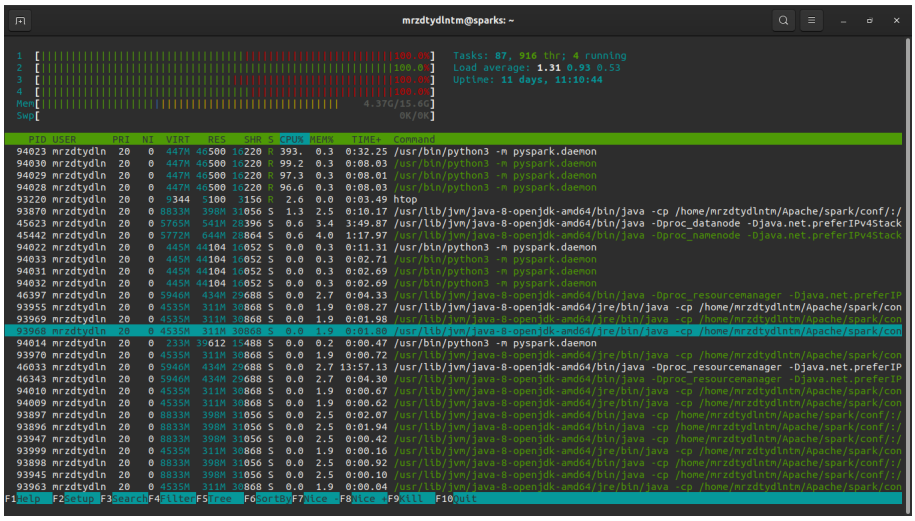
Lampiran 5: Kode Eksekusi dengan Threading

```
1 from huckel import huckel_2
2 import numpy as np
3 from numpy import linalg as la
4 import time
5 from scipy import linalg as la
6 import threading
7
8 def main():
9     start = time.time()
10    X = np.arange(6,101,2)
11    for x in X:
12        t1 = threading.Thread(target=huckel_2, args=(x,))
13        t1.start()
14        t1.join()
15    stop = time.time()
16    print('Waktu komputasi = {} s'.format(stop-start))
17
18 if __name__ == "__main__":
19    main()
```

Lampiran 6: *Process Load* dengan Multiprocessing

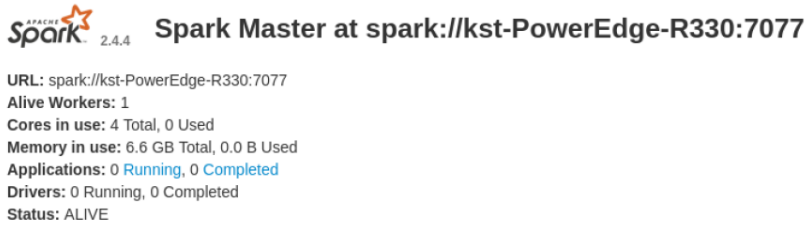


Lampiran 7: *Process Load* dengan PySpark



Lampiran 8: Spark Master pada Server Google Cloud

Lampiran 9: Spark Master pada Server KST



Lampiran 10: Spesifikasi Server Google Cloud