LAPORAN KERJA PRAKTEK

Simulasi Makroskopik untuk Dinamika Internal pada

Pemrosesan Nanomaterial dalam Ball Mill

di

Pusat Penelitian Fisika Lembaga Ilmu Pengetahuan Indonesia

Diajukan untuk memenuhi persyaratan kelulusan

Matakuliah TF4001 Etika Profesi dan Kerja Praktek

Oleh:

Fajar Fauzi Hakim / 13305053

PROGRAM STUDI TEKNIK FISIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI BANDUNG

2011

Lembar Pengesahan

SIMULASI MAKROSKOPIK UNTUK DINAMIKA INTERNAL PADA PEMROSESAN NANOMATERIAL DALAM BALL MILL

Di Pusat Penelitian Fisika Lembaga Ilmu Pengetahuan Indonesia

oleh :

Fajar Fauzi Hakim / 13305053

disetujui dan disahkan sebagai

Laporan Kerja Praktek

Tangerang Selatan, 14 Maret 2011

Kepala Grup Fisika Teoritik dan Komputasi Pusat Penelitian Fisika LIPI

Dr. Laksana Tri Handoko, Ph.D.

NIP: 196805071987121001

Abstraksi

Kerja Praktek dilaksanakan di Grup Fisika Teoritik dan Komputasi Pusat Penelitian Fisika Lembaga Ilmu Pengetahuan Indonesia, mulai tanggal 1 Juni 2010 sampai dengan tanggal 31 Juni 2010. Kerja praktek yang dilakukan adalah mengembangkan program simulasi untuk pemodelan dinamika internal pada proses penumbukan dalam ball mill sebagai canonical ensemble dengan menggunakan metode integrasi Monte Carlo pada bahasa pemrograman Python. Program simulasi ini merupakan pengembangan dari program sebelumnya yang telah menghasilkan keluaran yang memuaskan. Akan tetapi, terdapat beberapa hal yang masih dapat dioptimalkan.

Program simulasi baru dibuat berdasarkan model matematis pada paper dan program simulasi yang telah dibuat pada penelitian sebelumnya. Pada akhir kerja praktek, program simulasi baru yang mendukung komputasi paralel dan memiliki kinerja yang lebih baik telah berhasil dibuat. Akan tetapi program simulasi ini masih memiliki beberapa hal untuk diperbaiki.

Kata kunci: *comminution, Monte Carlo, ball mill, OpenMPI, Python, LIPI, GFTK*

KATA PENGANTAR

Alhamdu lillahi rabbil 'alamin. Atas rahmat-Nya penulis dapat menyelesaikan tugas kerja praktek. Atas rahmat-Nya pula penulisan laporan kerja praktek ini dapat diselesaikan sebagai salah satu prasyarat kelulusan sarjana Program Studi Teknik Fisika ITB. Karenanya adalah satu keniscayaan pula untuk menyebutkan nama-Nya saat membaca laporan ini.

Seluruh proses pelaksanaan kerja praktek ini baik dalam pelaksanaan di lapangan maupun dalam penulisan laporannya merupakan suatu proses belajar, yang meski tidak sempurna, namun memberi kesan yang mendalam. Penulis mendapatkan kesempatan untuk mencoba mengaplikasikan segala hal yang didapat di bangku kuliah pada kursi kerja. Penulis juga mendapatkan kesempatan untuk mempelajari ilmu baru yang belum pernah dipelajari sebelumnya. Dan di atas semua itu, pengalaman beraktifitas di luar daerah kenyamanan kampus memberikan penulis kesempatan untuk mengembangkan diri lebih lanjut. Sekali lagi, meski tidak sempurna, namun mudah-mudahan memberikan banyak manfaat.

Penulis mengucapkan terima kasih kepada rekan-rekan kerja yang telah membantu dan memberikan dukungan selama pelaksanaan kerja praktek dan penulisan laporan kerja praktek ini, yaitu:

1. Dr. Ir. Suyatman dan Bapak Budiono Kartohadiprodjo selaku dosen mata kuliah etika profesi dan kerja praktek yang telah banyak memberikan bimbingan dan petunjuk.
2. Bapak Prof. Laksana Tri Handoko sebagai dosen pembimbing selama kerja praktek di lapangan.
3. Gagus K. S., S.Si, beserta rekan-rekan anggota GFTK dan P2 Fisika lainnya yang telah memberikan pengetahuan dan wawasan baru.
4. Hendra, Iyon, Zulham, dan Wildan selaku rekan selama saya menumpang di mess kompleks LIPI G10.

Serta pihak-pihak lainnya yang tidak dapat penulis sebutkan satu per satu. Semoga Allah memberikan kebaikan yang lebih baik dari kebaikan yang telah rekan-rekan berikan kepada penulis. Penulis memohon maaf jika rekan-rekan mendapati hal-hal yang kurang berkenan pada diri penulis.

Laporan ini telah ditulis dan disusun dengan sebaik-baiknya, namun penulis sadar bahwa laporan ini masih belum memenuhi kriteria sempurna. Oleh karena itu, segala kritik dan saran untuk memperbaiki penulisan dan isi laporan ini akan penulis terima dengan lapang dada. Semoga isi dari laporan ini dapat memberikan manfaat yang signifikan, terutama kepada para pembaca. Terima kasih.

Bandung, Februari 2011

Penulis

DAFTAR ISI

Abstraksi i
DAFTAR ISI iv
BAB I 1
PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Lingkup 3
1.3. Tujuan 3
1.4. Sistematika Pembahasan 3
BAB II 5
PUSAT PENGEMBANGAN FISIKA LEMBAGA ILMU PENGETAHUAN INDONESIA 5
2.1. Sejarah dan Tinjauan Umum 5
2.2. Lingkup Pekerjaan 8
2.3. Deskripsi Pekerjaan 10
2.4. Jadwal Kerja 11
BAB III 13
TEORI-TEORI PENUNJANG KERJA PRAKTEK 13
3.1. Sains Komputasi 13
3.2. Bahasa Pemrograman Python 14
3.3. Integrasi Monte Carlo 16
3.4. Pseudo-random Number Generator 17
3.5. Parallel Computing 20
BAB IV 22
PELAKSANAAN KERJA PRAKTEK 22
4.1. Input 22
4.2. Proses Pengerjaan 23
4.2.1. Eksplorasi 23
4.2.1.1. Pemahaman Algoritma dan Bahasa Pemrograman 24
4.2.1.2. Pemahaman Model Matematis 24
4.2.1.3. Pemahaman Metode Simulasi 29
4.2.2. Pembuatan Program Simulasi 31
4.2.2.1. Pembuatan Pseudo-Random Number Generator 32

4.2.2.2. Pembuatan Program Utama 32
4.2.2.3. Pembuatan Algoritma Parallel Computing 36
4.2.3. Pelaporan 37
4.3. Pencapaian Hasil 37
BAB V 40
KESIMPULAN DAN SARAN 40
5.1. Kesimpulan 40
5.2. Saran 40
DAFTAR PUSTAKA 42

BAB I

PENDAHULUAN

1.1. Latar Belakang

Peran komputasi dalam perkembangan ilmu fisika baik terapan maupun teoritik semakin besar. Hal ini dipengaruhi oleh semakin tingginya tuntutan untuk menyelesaikan problem-problem matematis kompleks pada suatu model fisika yang tidak dapat diselesaikan melalui metode analitik secara lebih presisi. Begitu pentingnya peran komputasi sehingga disebut-sebut sebagai pilar ketiga dalam sains selain eksperimen dan teori. Lebih jauh lagi, teknologi komputasi adalah poin penting dalan era *theory-driven experiments*, yang berarti bahwa penelitian dan percobaan pada era tersebut tidak lagi sekedar bergantung pada *trial and error*, akan tetapi didukung dan digerakkan oleh teori-teori dasar yang menyokongnya.

Teknologi dan sains material adalah salah satu bidang yang membutuhkan bantuan komputasi dalam pengembangannya. Baik dalam pengembangan material bidang energi, bidang sensor, bidang bioinformatika, dan lainnya, trend teknologi material yang semakin mengarah pada material nano membutuhkan dukungan teoritik yang kuat. Pengembangan teknologi ini secara eksperimental tidaklah murah. Selain biaya pengadaan bahan percobaan, karakterisasi material pun acap kali menggunakan peralatan yang mahal dan langka. Karena itulah, pemodelan material serta desain material melalui metode teoritik yang didukung oleh teknologi komputasi menjadi sangat dibutuhkan. Namun, pemodelan dan desain material terutama dalam skala nano membutuhkan sumber daya komputasi yang sangat besar. Karenanya, para peneliti telah mengembangkan metode dan algoritma numerik untuk membantu memodelkan perilaku material secara lebih

efisien. Perangkat-perangkat lunak dan keras juga banyak dikembangkan untuk membantu sistem komputasi yang lebih baik, misalnya pengembangan *parallel computing* dan sistem komputer grid.

Indonesia yang memiliki sumber daya mineral yang cukup tinggi, karenanya banyak peneliti di Indonesia sadar bahwa teknologi pengolahan material patut dikuasai oleh bangsa Indonesia. Cukup banyak penelitian baik secara teoritik dan eksperimental mengenai teknologi material dan pengembangan sains material dilakukan di Indonesia meski masih jauh secara kuantitas dari yang dilakukan oleh bangsa-bangsa maju. Atas kesadaran ini pula, penelitian di Lembaga Ilmu Pengetahuan Indonesia (LIPI) khususnya di Pusat Penelitian Fisika (PPF) banyak yang mengarah pada pengembangan material secara eksperimental maupun teoritik. Salah satunya adalah penelitian mengenai pemodelan dinamika internal dalam proses *milling* yang dilakukan Grup Fisika Teoritik dan Komputasi (GFTK).

Proses *milling* merupakan salah satu metode mekanik untuk mendapatkan material berukuran nano secara *top-down*. Instrumen untuk proses *milling* ini dapat berupa *ball mill, roller mill, hammer mill,* dan lain-lain. *Ball mill* merupakan instrumen yang umumnya terdiri dari vial berisi bola-bola penghancur dengan gerakan yang dapat diatur, misalnya translasi atau sentrifugal. Meskipun instrumen *ball mill* dapat dikatakan sederhana parameter internal dan eksternal yang terlibat dalam proses *milling* ini cukup banyak. Seluruh parameter tersebut mengarah pada ketidakpastian yang menjadi salah satu kendala untuk melakukan pemodelan guna melakukan efisiensi proses *milling.* Model-model yang dibuat pada penelitian sebelumnya umumnya diturunkan dari persamaan gerak dan menghasilkan model yang terlalu rumit sehingga cukup lama untuk diselesaikan dan disimulasikan.

Penelitian yang dilakukan di GFTK LIPI [1] menghasilkan model dinamika internal pada proses penumbukan dalam ball mill. Pada model tersebut, dinamika

internal dijabarkan melalui pendekatan Hamiltonian yang menjelaskan interaksi-interaksi material di dalam *ball mill* saat proses penumbukan. Observabel fisik terhadap besaran termodinamik dilakukan melalui bantuan fungsi partisi dengan memandang sistem Hamiltonian sebagai *canonical ensemble* dalam mekanika statistik. Pada penelitian selanjutnya, analisa numerik dengan metode Monte Carlo dan simulasi untuk *ball mill* dengan model geometri vial *spex mill* dilakukan [2, 3].

Program simulasi yang telah dibuat pada penelitian sebelumnya masih dapat dioptimalkan dan ditambah dengan fungsi-fungsi tertentu. Pembuatan program simulasi yang lebih baik inilah yang dilakukan selama kerja praktek.

1.2. Lingkup

Lingkup kerja pada kerja praktek ini meliputi pembuatan program simulasi baru dengan bahasa Python. Program simulasi baru ini juga dicoba untuk dilengkapi dengan pemrograman paralel. Karena itu, lingkup kerja juga meliputi persiapan hal-hal yang diperlukan untuk menguji program pada lingkungan komputasi paralel.

1.3. Tujuan

Tujuan dari kerja praktek ini adalah:

1. Mempelajari proses analisis numerik untuk pembuatan program simulasi dari model matematis pemrosesan material.
2. Membuat program simulasi dinamika internal pada proses *milling* dalam *ball mill* yang menghasilkan output sesuai dengan penelitian sebelumnya namun lebih optimal.
3. Mengenal teknik komputasi fisika dengan metode *parallel computing*.

1.4. Sistematika Pembahasan

Laporan kerja praktek ini terdiri atas lima bab dengan susunan sebagai berikut:

- BAB I

  Pendahuluan, diuraikan latar belakang, lingkup, tujuan, dan sistematika pembahasan.

- BAB II

  Pandangan umum mengenai Pusat Penelitian Fisika LIPI, mencakup struktur organisasi, lingkup kerja, deskripsi kerja, dan jadwal.

- BAB III

  Teori dasar penunjang kerja praktek, meliputi teori dasar komputasi fisika, bahasa pemorograman Python, intergrasi Monte Carlo, Pseudo Random Number Generator, dan Parallel Computing.

- BAB IV

  Pelaksanaan kerja praktek, diuraikan input, tahap-tahap proses pengerjaan yaitu eksplorasi, pembuatan program simulasi, serta pelaporan, dan pencapaian hasil.

- BAB V

  Kesimpulan dan saran.

BAB II

PUSAT PENGEMBANGAN FISIKA LEMBAGA ILMU PENGETAHUAN INDONESIA

2.1. Sejarah dan Tinjauan Umum

Kegiatan kerja praktek dilaksanakan di lembaga PPF-LIPI yang berkantor di kompleks Pusat Penelitian Ilmu Pengetahuan dan Teknologi (Puspiptek) Serpong. Berikut ini adalah sejarah dan tinjauan umum dari PPF-LIPI serta Puspiptek Serpong yang bersumber dari Rencana Strategis LIPI Tahun 2010-2014 [4] dan laman-laman website lainnya yang memiliki keterkaitan dengan LIPI [5, 6, 7].

PPF-LIPI awalnya bernama Lembaga Fisika Nasional yang berdiri sejak 1967, kemudian nama lembaga tersebut berubah menjadi Pusat Penelitian dan Pengembangan Fisika Terapan (P3FT) berdasarkan Keputusan Presiden RI No. 1 tanggal 13 Januari 1986. PPF berada di bawah naungan Deputi Ilmu Pengetahuan dan Teknologi dan LIPI. PPF memiliki tugas untuk mendukung pembangunan nasional melalui riset pengembangan sumber daya alam terutama yang terkait dengan fisika. PPF-LIPI menerima kerjasama riset baik dari lembaga publik maupun mahasiswa yang ingin melakukan kerja praktek atau tugas akhir.

Gedung PPF-LIPI berada di komplek Puspiptek Serpong yang terletak di kota Tangerang Selatan, Banten. Pendirian Puspiptek Serpong pertama kali digagas oleh menteri riset Prof.Dr.Sumitro Djojohadikusumo pada 1976. Pada saat Menteri Riset dan Teknologi dijabat oleh Prof. Dr.-Ing. B.J. Habibie, gagasan tersebut mulai diimplementasikan. Saat ini Puspiptek memiliki 35 laboratorium yang telah beroperasi. Laboratorium-laboratorium tersebut secara teknis dikelola secara koordinatif antara LIPI, Badan Pengkajian dan Penerapan Teknologi

(BPPT), Badan Tenaga Atom Nasional (BATAN), dari Kementerian Riset dan Teknologi. Selain itu, terdapat dua laboratorium yang dikelola oleh Kementerian Lingkungan Hidup yaitu Sarana Pengendalian Dampak Lingkungan (Sarpedal), dan Pusdiklat Lingkungan.

Pusat Penelitian Fisika LIPI memiliki visi dan misi sebagai berikut:

- **Visi**

  Menjadi pusat unggulan dalam penelitian, pengembangan, dan penerapan ilmu pengetahuan dan teknologi berbasis fisika demi terwujudnya kehidupan bangsa yang berkeadilan, makmur, cerdas, kreatif, integratif dan dinamis.

- **Misi**

  a. Meningkatkan kompetensi inti ilmu pengetahuan dan teknologi berbasis fisika.
  b. Menumbuhkan, meningkatkan, dan mendayagunakan penelitian, pengembangan, dan penerapan serta rekayasa ilmu pengetahuan dan teknologi.
  c. Meningkatkan invensi dan inovasi di bidang ilmu pengetahuan dan teknologi.
  d. Meningkatkan deseminasi dan aplikasi ilmu pengetahuan dan teknologi untuk memperkuat daya saing industri dan ekonomi.
  e. Menyiapkan bahan untuk perumusan kebijakan di bidang ilmu pengetahuan dan teknologi.
  f. Mendukung terciptanya lingkungan yang berkwalitas dan berkelanjutan.

Tugas pokok dan fungsi dari Pusat Penelitian Fisika LIPI adalah sebagai berikut:

- **Tugas**

  Pusat Penelitian Fisika LIPI mempunyai tugas melaksanakan penyiapan bahan perumusan kebijakan, penyusunan pedoman, pemberian bimbingan teknis,

penyusunan rencana dan program, pelaksanaan penelitian bidang fisika serta evaluasi dan penyusunan laporan.

- **Fungsi**

  Untuk menyelenggarakan tugas tersebut, Pusat Penelitian Fisika LIPI mempunyai fungsi untuk:

  a. Penyiapan bahan perumusan kebijakan penelitian di bidang fisika.
  b. Penyusunan pedoman, pembinaan, dan pemberian bimbingan teknis penelitian bidang fisika.
  c. Penyusunan rencana, program dan pelaksanaan penelitian bidang fisika.
  d. Pemantauan pemanfaatan hasil penelitian bidang fisika.
  e. Pelayanan jasa ilmu pengetahuan dan teknologi bidang fisika.
  f. Pelaksanaan urusan tata usaha.

Di dalam Pusat Penelitian Fisika LIPI terdapat berbagai grup riset. 3 diantaranya adalah:

1. Grup Fisika Teoritik dan Komputasi (GFTK)

   Grup riset ini mengkaji ilmu dasar (teori) bidang fisika, komputasi, dan matematika.
2. Grup Optik Kuantum

   Grup Optik Kuantum merupakan kolaborasi riset peneliti teori dan terapan terkait dengan fenomena dan aplikasi laser di PPF. Kegiatan riset ini dilaksanakan dan dikelola oleh GFTK dan Kelompok Laser di PPF.
3. Grup Terahertz Photonics

   Grup riset ini mengkaji penerapan fiber optik dalam instrumentasi, sensor, dan penanganan bencana.

PPF memiliki 192 orang sivitas peneliti, tidak termasuk peneliti tamu. SDM peneliti diperoleh melalui program Penerimaan CPNS LIPI yang diselenggarakan tiap tahun. Untuk mendukung penelitian baik yang dilakukan oleh sivitas maupun

publik, PPF memiliki 2 laboratorium yang dapat dipergunakan yaitu Laboratorium Uji Tarik Mekanik dan Laboratorium Komputasi Terdistribusi.

Grup riset tempat penulis melaksanakan kerja praktek adalah GFTK. Pendirian grup ini pada 1 Januari 2001 menjadi tonggak awal mulai diperkenalkannya kajian teoritik sebagai salah satu kompetensi penelitian di LIPI dan PPF pada khususnya. Kepala GFTK adalah Dr. Laksana Tri Handoko, Ph.D. yang merupakan sivitas unggulan PPF-LIPI. GFTK ini juga yang mengelola Laboratorium Komputasi Terdistribusi.

2.2. Lingkup Pekerjaan

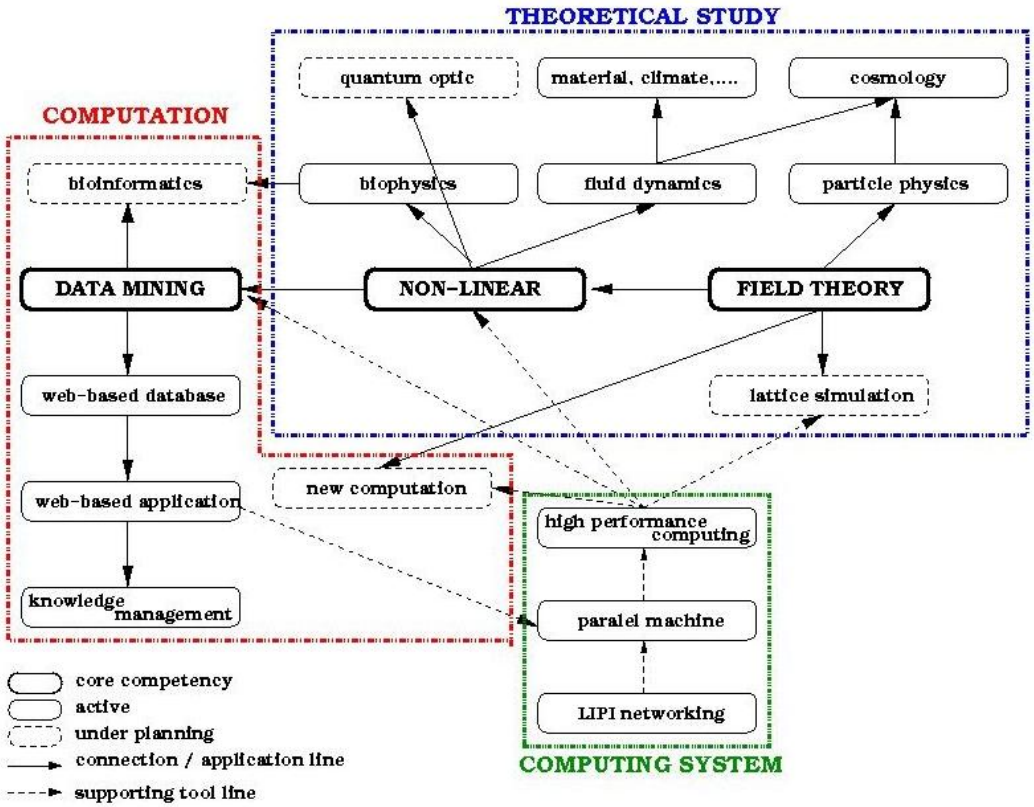
GFTK PPF-LIPI secara garis besar melakukan kajian fisika teoritik, komputasi, dan matematika terkait. Saat ini topik penelitian difokuskan pada :

1. **Fisika teori :**
   a. Fisika energi tinggi : teori unifikasi, neutrino, flavor physics.
   b. Fisika nonlinier : fluida berbasis teori medan, biofisika, plasma, kosmologi.

2. **Ilmu komputasi :**
   a. Pemodelan dan simulasi material : produksi nanomaterial, optik.
   b. Pemodelan dan simulasi biofisika : protein, DNA, nanofluidik.

3. **Informatika :**
   a. Model dan teknik penambangan data.
   b. Manajemen pengetahuan berbasis situs.
   c. Sistem informasi online berbasis situs.

Peta riset GFTK ditunjukkan pada Gambar 5.1 berikut ini.



Gambar 5.1 Peta riset GFTK [8, *http://sains.org/handoko/photo/roadmap_riset.jpg*]

GFTK dilengkapi dengan fasilitas sebagai berikut:

- Ruang kerja peneliti dan mahasiswa, serta ruang diskusi.
- Ruang kerja perangkat keras instrumentasi dan komputasi.
- Mesin komputasi paralel (*cluster computer*).
- Fasilitas jaringan komputer (LAN, server utama, *network printer*, dan hotspot).

Dalam aktifitasnya, grup riset ini kerap kali bekerja sama dengan pihak-pihak dan lembaga-lembaga lain baik lembaga internal di PPF, LIPI, lembaga ilmiah nasional, dan bahkan lembaga ilmiah internasional. Dalam kontribusi nya terhadap masyarakat, GFTK juga mengelola berbagai database dan portal ilmiah untuk publik. Aktifitas ilmiah di GFTK selain menghasilkan karya ilmiah juga mengeluarkan berbagai karya aplikatif baik berupa perangkat lunak maupun

perangkat keras. Di antaranya adalah LIPI Public Cluster dan toolkit-nya (openPC) dan LIPI Networked Robot dan arsitekturnya (openNR).

Saat penulis melaksanakan kerja praktek, sivitas aktif di GFTK sangat sedikit, yaitu sekitar 2-3 orang. Anggota-anggota GFTK saat itu umumnya sedang melanjutkan studi di luar negeri. Meskipun begitu, ruang GFTK masih sering menjadi tempat diskusi antara Dr. Laksana Tri Handoko, Ph.D. dengan kolega risetnya.

Salah satu hasil penelitian GFTK adalah pemodelan proses penumbukkan pada *ball mill* sebagai *canonical ensemble*. Salah satu bagian dari penelitian tersebut adalah pembuatan program simulasi untuk menghitung evolusi dari jumlah partikel dalam parameter observabel fisis berupa temperatur dengan menggunakan model instrumen *ball mill* tipe *spex mill*. Penelitian tersebut telah diterbitkan dalam Journal of Computational and Theoretical Nanoscience pada 13 Mei 2010. Meskipun begitu, program simulasi penghitungan tersebut, menurut kepala GFTK, masih dapat dioptimalkan lagi kualitasnya. Karenanya, penulis selaku peserta kerja praktek di GFTK, ditugaskan untuk melakukan optimasi dan pengembangan tersebut.

2.3. Deskripsi Pekerjaan

Deskripsi pekerjaan yang dilakukan selama kerja praktek di PPF adalah membangun program simulasi untuk pemodelan dinamika internal pada proses penumbukan dalam *ball mill* sebagai *canonical ensemble* dengan menggunakan metode integrasi Monte Carlo pada bahasa pemrograman Python. Program tersebut memiliki spesifikasi khusus berikut:

- Menghasilkan bilangan random dengan algoritma buatan sendiri agar nilai random yang dihasilkan dapat memenuhi ketelitian yang diharapkan.

- Melakukan integrasi Monte Carlo untuk menghasilkan nilai temperatur yang merepresentasikan evolusi jumlah partikel.
- Melakukan integrasi Monte Carlo dengan resolusi 10^8 x 10^8 x 10^8 agar diperoleh hasil dengan akurasi yang realistis (orde puluhan nanometer).
- Menghasilkan nilai output simulasi yang sesuai dengan program sebelumnya.
- Mengaplikasikan metode *parallel computing* pada program.

Dalam proses pengerjaan, penulis melakukan studi literatur, khususnya pada paper yang telah dibuat oleh peneliti sebelumnya dan sumber-sumber tambahan lainnya. Penulis juga melakukan diskusi dengan pembimbing kerja praktek yaitu kepala GFTK sendiri dan salah seorang sivitas GFTK yang terlibat dalam penelitian sebelumnya. Meskipun program simulasi sebelumnya sudah tersedia, namun penulis ditugaskan untuk menurunkan algoritma pemrograman sendiri dari model matematis yang dipaparkan pada paper.

2.4. Jadwal Kerja

Kerja praktek dilaksanakan mulai dari 1 Juni 2010 sampai dengan 30 Juni 2010. Waktu kerja peserta kerja praktek mengikuti waktu kerja sivitas PPF lainnya yaitu dari hari Senin sampai dengan Jumat, pukul 08.00 sampai dengan pukul 16.00 WIB dengan waktu istirahat dari pukul 12.00 – 13.00 untuk hari Senin – Kamis dan pukul 11.30 – 13.00 untuk hari Jumat.

Secara umum, kegiatan yang dilakukan selama kerja praktek adalah sebagai berikut:

1. Minggu pertama:
    - Pengenalan lingkungan kerja
    - Pengenalan teknik integrasi Monte Carlo dengan bahasa pemrograman Python
    - Pengenalan dan pembuatan *random number generator*

- Melakukan analisis kebutuhan algoritma program
- Pembuatan draft kasar program

2. Minggu kedua:
   - Penyempurnaan draft program
   - Pengenalan *parallel computing*
   - Persiapan dan pengaktifan fasilitas *cluster computer* di Laboratorium Komputasi Terdistribusi

3. Minggu ketiga:
   - Evaluasi program
   - Peningkatan kemampuan algoritma *random number generator*
   - Percobaan penggunaan algoritma *parallel computing*

4. Minggu keempat:
   - Re-evaluasi program
   - Koreksi bug dan kesalahan algoritma pada program
   - Percobaan penggunaan algoritma *parallel computing*

5. Minggu kelima:
   - Re-evaluasi program
   - Koreksi bug dan kesalahan algoritma pada program
   - Percobaan penggunaan algoritma *parallel computing*
   - Pembuatan final program

BAB III

TEORI-TEORI PENUNJANG KERJA PRAKTEK

3.1. Sains Komputasi

Seperti yang telah diterangkan pada BAB 1, komputasi telah menjadi bagian penting dalam pengembangan ilmu fisika dan juga penerapan fisika pada teknologi. Secara umum, sains dewasa ini sangat bergantung pada komputasi, bahkan ada ungkapan bahwa sains adalah sesuatu yang kita pahami dengan benar sehingga kita bisa 'menjelaskannya' pada komputer. Untuk menghasilkan data yang sesuai dengan kebutuhan secara efisien, diperlukan metode komputasi yang tepat yang erat kaitannya dengan pemrograman. Karenanya dalam hal ini, ilmuwan maupun insinyur memerlukan kemampuan untuk menulis program perangkat lunak, atau setidaknya mampu untuk mengubah code program yang sudah ada.

Dalam komputasi sains terdapat 3 tahap [9]:

- Pembangunan model; yang menghasilkan persamaan
- Simulasi; yang menghasilkan data
- Analisis

Model pada sains komputasi merupakan pengganti dari bahan eksperimen pada sains eksperimental. Pengujian pada sains eksperimental menghasilkan data untuk analisis, sedangkan sains komputasi menghasilkan data dari simulasi yang didasarkan pada model persamaan matematis. Model matematis pada sains komputasi harus dapat diuji kebenarannya. Model matematis ini menjadi pusat dari sains komputasi, karenanya hanya instruksi yang diformulasikan secara presisi yang dapat dieksekusi.

Model matematis tidak langsung dieksekusi pada tahap simulasi komputer. Aproksimasi perlu dilakukan agar komputasi dapat dilakukan dengan lebih efisien dan terkendali. Dari aproksimasi ini memang akan dihasilkan error, namun error yang terjadi merupakan error terkendali yang dapat diperkirakan dan dibuat sekecil mungkin. Dengan approksimasi ini, model matematis akan berubah menjadi model numerik atau komputasional, dan pada akhirnya, sains komputasi khususnya fisika komputasi intinya adalah mengembangkan metode aproksimasi untuk persoalan fisika dan teknik untuk mengimplementasikan metode aproksimasi tersebut pada komputer.

3.2. Bahasa Pemrograman Python

Python adalah bahasa pemrograman berorientasi objek yang dikembangkan pada akhir dekade 80an oleh Guido van Rossum sebagai bahasa pemrograman *scripting* di Centrum Wiskunde & Informatica (CWI), Amsterdam [9, 10]. Bahasa pemrograman ini didistribusikan secara gratis dan memiliki lisensi yang tidak mengikat.

Program yang dibuat dengan Python tidak dijalankan dengan kompilasi (*compile*) melainkan melalui *interpreter*. Ini membuat proses penulisan program dan menjalankannya lebih singkat dibandingkan bahasa pemrograman lainnya dan membuat pengguna dapat lebih berkonsentrasi pada prinsip dasar dari program yang ingin dibuat daripada pemrogramannya itu sendiri. Akan tetapi, program yang dibuat melalui *interpreter* Python ini tidak dapat menghasikan program *executable* yang berdiri sendiri sehingga hanya bisa dijalankan pada komputer yang telah dipasangi *interpreter* Python.

Python memiliki keunggulan-keunggulan yang membuatnya lebih mudah dipelajari dibanding bahasa pemrograman lain, antara lain [10]:

1. Python adalah perangkat lunak open-source, dengan kata lain, gratis. Python juga umumnya telah disertakan pada distribusi-distribusi LINUX.
2. Python tersedia untuk seluruh *operating system* utama (Linux,Unix,Windows, MacOS, dll.). Program yang ditulis pada satu sistem dapat dijalankan pada sistem lain tanpa harus melakukan pengubahan *code*.
3. Python lebih mudah dipelajari dan menghasilkan *code* yang lebih mudah dibaca daripada bahasa pemrograman lain.
4. Python dan perangkat-perangkat ekstensinya mudah dipasang.

Python sedikit banyak dipengaruhi oleh bahasa pemrograman Java dan C++ namun memiliki kemiripan pula dengan bahasa Matlab. Meskipun Python banyak mengadopsi konsep pemrograman berorientasi objek seperti *class* dan *methods*, pada penelitian ini Python akan diperlakukan sebagai pemrograman prosedural. Contoh program yang ditulis dengan Python adalah sebagai berikut:

```
def test2(i,j):
      for n in range(i,j):
            t=2
            r='1'
            for w in range(2, n):
                  if n % w == 0:
                        r=r+' '+str(w)
                        t+=1
            if t==2:
                  print n, 'adalah bilangan prima'
            elif t>2: print n,'memiliki ',t,' buah faktor: ',r,n

x=input('range bawah: ')
y=input('range atas: ')

test2(x,y)
```

Program di atas merupakan program untuk menampilkan faktor dari bilangan-bilangan pada *range* tertentu. Program ini juga memberitahukan apabila ditemukan suatu bilangan prima pada *range* bilangan tersebut.

3.3. Integrasi Monte Carlo

Monte Carlo merupakan metode komputasi yang memanfaatkan simulasi variabel acak untuk menyelesaikan persoalan fisika/matematika yang terlalu rumit untuk diselesaikan secara analitik maupun numerik. Dasar dari Monte Carlo ini adalah teori probabilitas dan statistik, khususnya teorema limit tengah pada statistika. Probabilitas adanyasuatu jumlah nilai-nilai identik dari variabel acak yang kontinu $\rho_{N=}\ \xi_{1+}\ \xi_{2+\ldots+}\ \xi_{N}$ pada interval (x', x") untuk nilai N yang besar pada sumbu x adalah:

$$P\{x' < \rho_N < x"\} \approx \int_{x'}^{x"} þ_{N}(x)dx \qquad (3.1)$$

Hal ini menunjukkan bahwa jumlah ρ_N dari sejumlah besar variabel acak yang identik dan independen dapat diaproksimasi normal dan dapat direpresentasikan sebagai grafik normal Gaussian.

Integrasi dengan metode Monte Carlo mempermudah penghitungan integrasi lipat banyak seperti berikut:

$$I = \int_{a1}^{b1} dx_1 \int_{a2}^{b2} dx_2 \ldots \int_{an}^{bn} dx_n f(x_1, x_2, \ldots, x_n) \equiv \int_V f(\overline{\boldsymbol{x}}) d\overline{\boldsymbol{x}} \qquad (3.2)$$

Dengan $\overline{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ dan V adalah volume integrasi, dengan kata lain $V = (b1 - a1) \times (b2 - a2) \times \ldots \times (bn - an)$. Algoritma Monte Carlo biasa mengambil nilai uji secara uniform dari daerah integrasi untuk menghitung estimasi nilai integral dan errornya. Misalkan nilai-nilai uji berjumlah N dan setiap nilaiuji ditunjukkan oleh$\overline{x}_1, \ldots, \overline{x}_N$, maka estimasi nilai integral adalah [11]:

$$I \equiv V \frac{1}{N} \sum_{i=1}^{N} f(\overline{x}_i) \qquad (3.3)$$

Integrasi Monte Carlo adalah bentuk penggunaan paling dasar dari metode Monte Carlo. Berbagai pengembangan metode Monte Carlo telah dilakukan untuk kasus-kasus yang beragam seperti Quantum Monte Carlo, Metropolis Monte Carlo, dll.

3.4. Pseudo-random Number Generator

Metode integrasi Monte Carlo membutuhkan varibel acak dalam penghitungannya. Kualitas *randomness* dari varibel acak tersebut menentukan presisi dan akurasi hasil penghitungan integrasi Monte Carlo. Untuk mendapatkan variabel acak, terdapat 2 cara. Yang pertama adalah mengambil data dari fenomena fisika yang menghasilkan data acak, misalnya peluruhan bahan radioaktif. Yang kedua adalah melalui algoritma pemrograman yang sering disebut sebagai *pseudo-random number generator*.

Kelebihan variabel acak yang berasal dari data fenomena fisika adalah nilainya benar-benar acak, namun jika diperlukan bilangan acak dalam jumlah sangat banyak, maka akan dibutuhkan media penyimpanan data yang sangat besar. Ini membuatnya kurang fleksibel dan praktis untuk kebutuhan operasi yang melakukan penghitungan dengan bilangan random berkali-kali. Di lain pihak, *pseudo-random number generator* memiliki tingkat keacakan yang terbatas, tergantung metode algoritma yang digunakan. Karena hanya berupa algoritma pemrograman, bilangan acak dapat dihasilkan secara fleksibel dan dapat digunakan dimanapun menggunakan komputer. Karena itulah, *pseudo-random number generator* ini lebih dipilih untuk melakukan perhitungan-perhitungan yang membutuhkan variabel acak seperti perhitungan metode Monte Carlo.

Suatu *pseudo-random number generator* yang baik untuk kebutuhan penghitungan dalam pemodelan fisika pada masa kini perlu memenuhi kriteria-kriteria sebagai berikut [11]:

1. Distribusi yang baik. Titik-titik bilangan yang dihasilkan harus terdistribusi sesuai dengan distribusi yang terdapat pada bilangan yang benar-benar acak. Lebih jauh lagi, titik-titik yang dihasilkan oleh suatu *pseudo-random number generator* tidak boleh mengandung keterkaitan antara satu dengan yang lain.
2. Periode yang panjang. Setiap *pseudo-random number generator* memiliki periode; setelah menghasilkan sejumlah bilangan acak, ia akan menghasilkan bilangan-bilangan yang sama berulang-ulang. Untuk menghindari korelasi yang tidak diinginkan, kita harus menghasilkan bilangan acak jauh di bawah nilai periode tersebut saat melakukan kalkulasi.
3. *Repeatability* atau keberulangan. Untuk pengujian dan pengembangan, mungkin adalah suatu keharusan untuk mengulang suatu perhitungan dengan bilangan acak yang benar-benar sama dengan perhitungan sebelumnya. Lebih jauh lagi, *pseudo-random number generator* harus memungkinkan unruk mengulang sebagian dari suatu pekerjaan tanpa melakukan keseluruhan dari pekerjaan tersebut. Ini membutuhkan kemampuan untuk menyimpan suatu keadaan dari generator.
4. Memiliki sub-sekuen dengan nilai-nilai yang terpisah jauh. Untuk perhitungan yang sangat besar, akan sangat memudahkan jika dapat melakukan sub-simulasi secara independen yang hasilnya dapat digabungkan dengan mengasumsikan bilangan yang dihasilkan independen secara statisitik.
5. Portabilitas. Ini tidak hanya berarti bahwa rutin/kode untuk generator harus portabel (ditulis dalam bahasa pemrograman tingkat tinggi seperti Fortran dan C), tapi juga harus menghasilkan rangkaian bilangan yang sama pada mesin yang berbeda.
6. Efisien. Pembangkitan bilangan *pseudo-randpm* harus tidak terlalu memakan waktu. Hampir semua generator dapat diimplementasikan dengan cara yang efisien.

Selain hal-hal di atas, terdapat beberapa hal tambahan yang perlu diperhatikan untuk membuat *pseudo-random number generator* yang tepat untuk kebutuhan perhitungan. Diantaranya adalah:

1. Semakin banyak jumlah bilangan acak yang dimasukkan dalam perhitungan, hasilnya semakin mendekati teori.
2. Jumlah bilangan acak yang dihasilkan mencukup kebutuhan resolusi dari pemodelan fisika; misalkan pemodelan fisika membutuhkan ketelitian hingga ukuran mikrometer dalam tiga dimensi, maka dibutuhkan bilangan acak sebanyak 10^6x10^6x10^6.
3. Jumlah angka penting dibatasi; misalkan jika model fisika hanya membutuhkan angka penting sebanyak 2 angka di belakang koma, maka bilangan acak yang dihasilkan juga harus memiliki angka penting 2 angka di belakang koma, sebab jika lebih dari itu, 2 bilangan acak akan dianggap sama (1,234 dan 1,235 akan dianggap sama jika model fisika hanya mengambil 2 angka di belakang koma). Kriteria ini akan membatasi jumlah bilangan acak yang dapat dihasilkan pada *range* bilangan yang terbatas, karenanya perlu optimalisasi random variable agar sesuai dengan kebutuhan pemodelan fisika.

Salah satu algoritma *pseudo-random number generator* yang sering digunakan dalam berbagai bahasa pemrograman adalah Linear Congruential Method (LCM). Algoritma ini merupakan algoritma variabel acak yang sederhana. LCM menghasilkan bilangan acak melalui persamaan berikut:

$$X_{n+1} = (aX_n + c)\%m \tag{3.4}$$

Ketentuan-ketentuan untuk setiap parameter pada persamaan di atas adalah:

- m, *modulus*, $0 < m$
- a, *multiplier*, $0 < a < m$
- c, *increment*, $0 \leq c < m$
- X_0, *seed* (nilai awal), $0 \leq X_0 < m$
- $a\%m = a\%c = 1$

- c dan m merupakan bilangan prima relatif
- $a - 1$ dapat dibagi oleh faktor prima dari m
- $a - 1$ merupakan bilangan kelipatan 4 jika m juga kelipatan 4
- a harus sangat besar

Nilai m akan menentukan jumlah bilangan acak yang dihasilkan. LCM memang bukan algoritma penghasil variabel acak yang baik, namun pemilihan variabel $a, m,$ dan c dapat membantu meningkatkan keacakan dari bilangan yang dihasilkan. Meskipun berbagai nilai variabel-variabel tersebut telah diaplikasikan pada berbagai bahasa pemrograman, secara umum, LCM tidak begitu bagus untuk penghitungan metode Monte Carlo.

Metode *pseudo-random number generator* lainnya memiliki tingkat *randomness* yang lebih baik dan lebih cocok untuk penghitungan metode Monte Carlo. Beberapa diantaranya adalah Levitson-Sobol, Wichmann-Hill, dan Mersenne-Twister. Algoritma Mersenne-Twister merupakan algoritma *pseudo-random* default pada Python.

3.5. Parallel Computing

Parallel computing merupakan konsep metode komputasi menggunakan resource dari komputer-komputer terdistribusi dalam jaringan secara simultan untuk melakukan suatu operasi komputasi tertentu. Sistem ini tergolong sederhana dan merupakan salah satu cara murah dan mudah untuk meningkatkan performa komputasi tanpa harus membeli server atau workstation yang sangat mahal. *Parallel computing* dapat menjadi satu-satunya cara untuk melakukan komputasi pada komputasi yang membutuhkan kecepatan komputasi skala *teraflop* [12]. Kebutuhan akan *parallel computing* juga didorong dengan semakin beredarnya prosesor dengan dua inti atau lebih. *Parallel computing* dapat memaksimalkan kinerja penggunaan seluruh inti prosesor secara simultan.

Pada sistem *parallel computing* umumnya diperlukan antar muka komunikasi untuk menghubungkan prosesor-prosesor yang terhubung pada sistem *parallel computing*. Contoh perangkat lunak antar muka komunikasi yang umum digunakan adalah MPI, OpenMP, MPICH, dll. Untuk dapat mengakses antar muka tersebut dari program yang kita buat diperlukan antar muka atau library tambahan, misalnya untuk mengakses antar muka MPI pada bahasa Python, diperlukan PyMPI.

BAB IV
PELAKSANAAN KERJA PRAKTEK

4.1. Input

Program simulasi dibuat berdasarkan paper penelitian [2] berjudul '*Modelling Comminution Processes in Ball Mills as a Canonical Ensemble*' yang ditulis oleh G. K. Sunnardianto, Muhandis, F. N. Diana, dan L. T. Handoko. Penulis pertama paper tersebut, G. K. Sunnardianto, merupakan programmer yang membuat program simulasi sebelumnya pada paper tersebut dan merupakan narasumber utama dalam pengerjaan tugas kerja praktek. Sebenarnya aplikasi yang telah beliau buat dapat dijadikan rujukan untuk pengerjaan tugas kerja praktek yang dilakukan penulis, namun Bapak L. T. Handoko selaku pembimbing kerja praktek menginginkan agar penulis menurunkan sendiri algoritma program dari model matematis yang dijelaskan pada paper. Baru setelah hal tersebut dilakukan dan program simulasi buatan penulis masih tidak mendapatkan hasil yang sama dengan program simulasi sebelumnya, penulis menjadikan program simulasi sebelumnya tersebut sebagai rujukan untuk kemudian dicari peluang-peluang optimasi yang dapat dilakukan. Diskusi juga dilakukan dengan pembimbing untuk mengevaluasi program yang penulis buat dan mengatasi bug serta permasalahan yang muncul.

Untuk mempelajari *parallel computing* khusunya infrastruktur LIPI Public Cluster yang tersedia di tempat kerja praktek, penulis melakukan diskusi melalui surat elektronik dengan Bapak Zainal Akbar, sivitas GFTK yang penelitiannya berfokus pada *parallel computing*. Diskusi jarak jauh tersebut dilakukan karena beliau adalah yang paling memahami *parallel computing* sedangkan beliau sedang melanjutkan studi di Jerman. Selain itu, pada saat pelaksanaan kerja praktek, tidak ada sivitas yang benar-benar menguasai *parallel computing* di PPF. Bapak

Bambang Hermanto, pengelola jaringan komputer di PPF-LIPI, juga turut membantu penulis dalam mengoperasikan LIPI Public Cluster meskipun dalam kewenangan terbatas.

Beberapa paper lain [1], [13], dan sumber literatur lainnya seperti buku dan artikel-artikel di internet juga digunakan untuk membantu memahami algoritma yang perlu dibuat. Paper dan manual lainnya [14,15] juga dipelajari untuk memahami pemrograman parallel khususnya dalam bahasa pemrograman Python. Ketersediaan hotspot dan akses internet memfasilitasi penulis untuk mencari sumber rujukan lainnya. Untuk mencetak dokumen, disediakan pula printer yang terhubung dengan jaringan komputer.

4.2. Proses Pengerjaan

Proses pengerjaan dapat dibagi menjadi tiga tahap: eksplorasi, pembuatan program simulasi, dan pelaporan hasil kerja praktek.

Pada LIPI Public Cluster yang terdapat di Laboratorium Komputasi Terdistribusi telah terinstal antar muka OpenMPI dan PyMPI. Selain itu, konsep *parallel computing* di LIPI telah dikembangkan menjadi suatu *grid* komputer yang dapat dipergunakan oleh publik dan dimonitor penggunaannya melalui antar muka web. Namun saat kerja praktek dilaksanakan, LIPI Public Cluster sedang ditutup untuk umum dan hanya dibuka untuk keperluan riset sivitas GFTK. Meskipun demikian, sivitas GFTK jarang mempergunakannya sehingga pada saat kerja praktek hampir seluruh *cluster* komputer dimatikan dan tidak dipasangi kabel jaringan.

4.2.1. Eksplorasi

Tahap eksplorasi meliputi pemahaman algoritma bahasa pemrograman, pemahaman model matematis, dan pemahaman metode simulasi.

4.2.1.1. Pemahaman Algoritma dan Bahasa Pemrograman

Proses ini dimulai dengan memperdalam penguasaan pemrograman bahasa Python dan mempelajari pembuatan algoritma untuk melakukan penghitungan integrasi Monte Carlo serta pembuatan *pseudo-random number generator*. Kedua hal tersebut merupakan hal yang esensial untuk pembuatan program simulasi. Untuk hal tersebut, rujukan [16] sangat membantu. Beberapa rujukan lain [17], [18], dan [19] juga digunakan terutama untuk menemukan metode algoritma yang tepat untuk proses iterasi pada program simulasi dan pembuatan algoritma *pseudo-random number generator*.

4.2.1.2. Pemahaman Model Matematis

Langkah selanjutnya adalah mempelajari model matematis dari penumbukan ball mill pada paper [2]. Bagian paling penting adalah model matematis final dalam penelitian tersebut yang juga telah disimulasikan pada program sebelumnya. Selain model matematis final, penurunan persamaan juga penting untuk dipelajari agar lebih memahami model matematis tersebut. Dari hal-hal tersebut, dapat diketahui input dan output dari program simulasi dan informasi-informasi untuk membangun algoritma pada program seperti jumlah dan posisi operasi iterasi, fungsi yang perlu digunakan, dan lain-lain.

Persamaan untuk memodelkan penumbukan dalam ball mill umumnya melibatkan *physical observable* berupa ukuran bulir partikel dalam *equation of motions* (EOMs). Namun, cara ini memiliki kekurangan-kekurangan sebagai berikut [1, 2]:

- Secara eksperimen, mustahil untuk melacak perpindahan geometris dari materi-materi di dalam tabung ball mill dengan resolusi waktu yang sesuai. Masalah ini semakin buruk jika simulasi dilakukan pada skala nanometer dengan ukuran resolusi ruang-waktu yang sebanding.
- Menyelesaikan EOMs secara numerik dan melakukan simulasi dengan tingkat akurasi yang tinggi (resolusi waktu yang cukup) membutuhkan kapasitas komputasi yang besar dan waktu *running* simulasi yang lama.

Pendekatan lain yang lebih empirik untuk memodelkan proses penumbukan dalam ball mill didasarkan pada pemodelan yang lebih realistik secara fisika. Pendekatan ini secara umum meliputi 3 aspek:

1. Evaluasi dinamika badan milling dan input energi ke dalam bubuk.
2. Deskripsi pengaruh input energi tersebut terhadap pecahnya bubuk.
3. Deskripsi evolusi bulir pada bubuk dalam parameter distribusi ukuran partikel.

Pemodelan cara baru ini menggunakan pendekatan *hamiltonian* untuk memodelkan seluruh interaksi yang relevan secara empirik. Hasilnya akan terbangun persamaan total *hamiltonian* yang menjelaskan dinamika di dalam ball mill. Setelah itu, *thermodinamical observables* diekstrak melalui fungsi partisi dengan memperlakukan sistem sebagai *canonical ensemble* pada temperatur yang terbatas. Dengan pendekatan ini juga parameter eksternal dari lingkungan di sekitar *vial* ball mill, seperti medan elektromagnetik dan lainnya, dapat dimasukkan dalam perhitungan.

Dinamika material pada vial, yaitu bubuk (*powder*) dan bola (*ball*), direpresentasikan dengan *hamiltonian* $H_{\mathrm{m}}(\vec{r},t)$ dengan indeks m merupakan notasi untuk material yang berupa p (*powder*) atau b (*ball*) dan $\vec{r}$ merupakan vektor posisi material dalam kordinat Cartesian (x,y,z). *Hamiltonian* merepresentasikan interaksi-interaksi yang relevan antar material di dalam sistem sebagai berikut [1]:

$$H_m = H_0 + V_{m-m} + V_{m-v} + V_{m-m'} + V_{ext} \tag{4.1}$$

Huruf v merupakan indeks untuk vial dan H_0 adalah *hamiltonian* materi bebas yang mengandung representasi energi kinetik,

$$H_0 = \frac{1}{2m_m}\sum_{i=1}^{n_m}|(\vec{p}_m)_i|^2 \tag{4.2}$$

dengan n_m untuk jumlah material, m_m untuk massa material, dan $\vec{p}_m$ untuk momentum material. Massa dan evolusi ukuran material diasumsikan sama untuk jenis material yang sama.

Interaksi antar sesama material sejenis (V_{m-m}), antara material dengan dinding vial (V_{m-v}), dan antar material berlainan jenis ($V_{m-m'}$) dapat direpresentasikan melalui misalnya dengan potensial tumbukan (V^{imp}) yang memiliki persamaan sebagai berikut:

$$V_{m-m'}^{imp}(\vec{r},t) = -\sum_{i=1}^{n_m}\sum_{j=1}^{n_{m'}}\int_0^{(\xi_{mm'})_{ij}} d\,(\xi_{mm'})_{ij}\vec{n}\,.\,(\vec{F}_{mm'}^{imp})_{ij} \qquad (4.3)$$

dengan m dan m' dapat berupa b, p, atau v serta $\vec{n}$adalah vektor normal satuan. Potensial harus merepresentasikan seluruh dinamika mekanika klasik pada material, dalam hal ini, gaya tumbukan pada *ball* dan *powder*. Gaya tumbukan didominasi oleh komponen-komponen normalnya,

$$\vec{F}_{m-m'}^{imp}(\vec{r},t) = \left[\frac{2\Upsilon_{mm'}}{3(1-v_{mm'}^2)}\sqrt{R_{mm'}^{eff}}\left(\xi_{mm'}^{3/2} + \frac{3}{2}A_{mm'}\sqrt{\xi_{mm'}}\frac{d\xi_{mm'}}{dt}\right)\right]\vec{n} \quad (4.4)$$

$\Upsilon_{mm'}$adalah modulus Young material, $v_{mm'}$adalah rasio Poisson material, $R_{mm'}^{eff}$ adalah jari-jari efektif antar material yang dihitung dengan persamaan $R_{mm'}^{eff} = \frac{R_m R_{m'}}{R_m + R_{m'}}$, dan $\xi_{mm'}$ adalah perpindahan yang dihitung dengan persamaan $\xi_{mm'} = R_m + R_{m'} - |\vec{r}_m - \vec{r}_{m'}|$, R_m adalah jari-jari material yang berinteraksi. A adalah parameter dissipatif yang mengandung konstanta viskositas η_m. Potensial-potensial lainnya seperti potensial Coulomb dan gravitasi juga dapat mempengaruhi material di dalam sistem. Potensial-potensial tersebut dapat diabaikan karena jarak antar material yang sangat pendek dan massa material yang sangat kecil.

Jika pengaruh medan elektromagnetik eksternal dimasukkan dalam perhitungan, maka persamaan (4.2) dapat ditulis sebagai berikut:

$$H_{0+EM} = \frac{1}{2m_m}\sum_{i=1}^{n_m}\left|(\vec{p}_m)_i - Q_m\vec{A}\right|^2 + n_m Q_m \phi \tag{4.5}$$

dengan ϕ dan $\vec{A}$ adalah potensial elektromagnetik skalara dan vektor serta Q_m adalah muatan dari material.

Pada proses penumbukan, hal yang menjadi perhatian utama adalah dinamika dari *powder*. Oleh karena itu, total *hamiltonian* untuk *powder* dapat dituliskan sebagai berikut:

$$\begin{aligned} H_p = {} & \frac{1}{2m_p}\sum_{i=1}^{n_p}\left|\left(\vec{p}_p\right)_i - Q_p\vec{A}\right|^2 + n_p Q_p \phi \\ & -\frac{1}{2}\sum_{i(\neq j)=1}^{n_p}\sum_{j=1}^{n_p}\int_0^{(\xi_{\mathrm{pp}})_{\mathrm{ij}}} d\,(\xi_{\mathrm{pp}})_{\mathrm{ij}}\vec{\mathrm{n}}\,.\left(\vec{\mathrm{F}}_{\mathrm{pp}}^{\mathrm{imp}}\right)_{\mathrm{ij}} \\ & -\sum_{m:b,v}\sum_{i=1}^{n_p}\sum_{j=1}^{n_m}\int_0^{(\xi_{\mathrm{pm}})_{\mathrm{ij}}} d\,(\xi_{\mathrm{pm}})_{\mathrm{ij}}\vec{\mathrm{n}}\,.\left(\vec{\mathrm{F}}_{\mathrm{pm}}^{\mathrm{imp}}\right)_{\mathrm{ij}} \end{aligned} \tag{4.6}$$

untuk $Q_p \neq 0$. Kedua suku potensial terakhir dalam persamaan di atas merepresentasikan total tumbukan antar *powder*, antara *powder* dengan *ball*, dan antara *powder* dengan *vial*. Di sini interaksi antar *ball* dan antara *ball* dengan *vial* tidak perlu diperhitungkan. Inilah keuntungan dari metode *hamiltonian*.

Selanjutnya model matematis diturunkan dengan menghubungkan persamaan *hamiltonian* yang sudah terbentuk dengan observabel fisik makroskopik. Hal ini dilakukan melalui fungsi partisi pada mekanika statistik,

$$Z_m = \int \prod_{i=1}^{n_m} d\vec{p}_i d\vec{r}_i \; exp\left[\int_0^{\beta} dt \; H_m\right] \tag{4.7}$$

untuk suatu *canonical ensemble* dari material m yang diatur oleh *hamiltonian* H_m. β dalam persamaan tersebut didefinisikan sebagai $\beta \equiv 1/(k_B T)$ dengan k_B merupakan konstanta Boltzman dan T merupakan temperatur absolut. Selanjutnya, tekanan ternormalisir dapat dihitung sebagai berikut:

$$P'_m = \frac{\ln Z_m}{\ln Z_{0m}} \tag{4.8}$$

Dengan mengintegrasikan komponen waktu dari persamaan di atas pada temperatur terbatas, maka akan didapat persamaan berikut:

$$P'_p = 1 - \beta\mathcal{F} \ln^{-1}\left(\frac{1 m_p \pi}{\beta}\right) \tag{4.9}$$

dengan $\mathcal{F}$ merupakan hasil dari persamaan berikut:

$$\begin{aligned}\mathcal{F} \; = 2 \int \prod_{i=1}^{n_p} d\vec{r}_i \Bigg[& Q_p \phi - \frac{2}{15 n_p} \sum_{i(\neq j)=1}^{n_p} \sum_{j=1}^{n_p} \frac{2\Upsilon_{pp}}{1 - v_{pp}^2} \sqrt{R_{pp}^{eff}} (\xi_{pp})_{ij}^{5/2} \\ & - \frac{4}{15 n_p} \sum_{m:b,v} \sum_{i=1}^{n_p} \sum_{j=1}^{n_m} \frac{2\Upsilon_{pm}}{1 - v_{pm}^2} \sqrt{R_{pm}^{eff}} (\xi_{pm})_{ij}^{5/2} \Bigg]\end{aligned} \tag{4.10}$$

Persamaan (4.9) menunjukkan perilaku umum dalam model mengenai nilai tekanan yang bergantung pada temperatur. Struktur geometris dan pergerakan *vial* yang merupakan salah satu input terpenting dalam model matematis ini berperan di dalam fungsi $\mathcal{F}$.

Persamaan (4.10) merupakan hasil akhir yang siap untuk dievaluasi lebih lanjut dengan simulasi Monte Carlo.

4.2.1.3. Pemahaman Metode Simulasi

Seperti yang sudah disebutkan sebelumnya, model matematis yang disimulasikan adalah fungsi $\mathcal{F}$ yang akan menunjukkan perilaku dari ukuran dan jumlah dari material dalam *vial* pada proses penumbukan. Simulasi dilakukan dengan mengasumsikan seluruh material adalah bola-bola yang saling berinteraksi. Model ball mill yang digunakan pada simulasi ini adalah *spex mill*. Program simulasi dapat juga digunakan untuk model ball mill lainnya dengan mengubah parameter koordinat yang digunakan.

Pada *spex mill*, sistem terdiri dari *vial* yang bergerak dalam koordinat non-inersial dengan struktur geometri seperti pada gambar 1 dan 2 dari [13]. Karena sistem *spex mill* ini bergerak pada koordinat non-inersial $\vec{r}$ = (*x*,*y*,*z*), maka sistem koordinat pada model matematis harus diubah menjadi sistem koordinat inersial (*X*,*Y*,*Z*). Untuk melakukan transformasi koordinat tersebut, digunakan transformasi roto-translasi sebagai berikut [13]:

$$X(t) = x\cos\theta(t)\cos\alpha(t) + y\cos\theta(t)\sin\alpha(t) + z\sin\alpha(t) + L\sin\theta(t) \quad (4.11)$$

$$Y(t) = -x\sin\alpha(t) + y\cos\alpha(t) \quad (4.12)$$

$$Z(t) = -x\sin\theta(t)\cos\alpha(t) - y\sin\theta(t)\sin\alpha(t) + z\cos\alpha(t) + L\cos\theta(t) \quad (4.13)$$

dengan L adalah panjang batang lengan mekanik dari *spex mill*, θ adalah sudut yang memiliki sumbu rotasi berupa sumbu *Y* pada koordinat inersial, dan α adalah sudut yang memiliki sumbu rotasi berupa sumbu *z* pada koordinat non-inersial. Kedua sudut tersebut dapat ditulis sebagai berikut:

$$\theta = \theta_0 \sin(\omega t + \varphi) \quad (4.14)$$

$$\alpha = \alpha_0 \sin(\omega t + \varphi) \quad (4.15)$$

dengan θ_0 dan α_0 merupakan momentum sudut pada masing-masing sumbu, ω merupakan frekuensi sudut, dan φ merupakan faktor fasa yang bergantung pada kondisi awal. Untuk melakukan simulasi pada model ball mill selain *spex mill* yang perlu dilakukan hanyalah mengganti persamaan transformasi roto-translasi pada persamaan (4.11) ~ (4.13) sesuai dengan pergerakan geometri dari ball mill.

Simulasi metode Monte Carlo dilakukan dengan membangkitkan sejumlah bilangan random untuk menyimulasikan titik-titik posisi dari *powder* dan *ball* pada batas koordinat yang ditentukan oleh geometri *vial*. Simulasi *powder* dan *ball* yang dihasilkan dijadikan input untuk model matematis yang dihitung yaitu persamaan (4.10). Pada simulasi akan dilakukan perhitungan integrasi 3 dimensi yang akan diselesaikan dengan metode integrasi Monte Carlo. Resolusi perhitungan yang dibutuhkan bergantung pada ukuran dari *powder*. Sebagai contoh, untuk ukuran *powder* sebesar 100μm diperlukan resolusi minimal sebesar ~ 10^4 x 10^4 x 10^4 atau 10^{12}. Hal ini tentunya akan membuat waktu simulasi sangat lama sehingga membutuhkan resource CPU yang sangat besar. Karenanya, sistem *parallel computing* cukup penting untuk diimplementasikan pada program simulasi. Untuk hal tersebut, penulis mempelajari rujukan [14] dan [15].

Sebelum dijadikan input pada persamaan (4.10), nilai dari titik-titik *powder* dan *ball* yang dibangkitkan harus diuji terlebih dahulu untuk menentukan apakah terjadi interaksi antar material atau tidak. Hal ini diperlukan karena persamaan (4.10) hanya memperhitungkan material yang berinteraksi sedangkan titik-titik *powder* dan *ball* yang dibangkitkan belum tentu menghasilkan interaksi antar material. Interaksi yang diperhitungkan hanyalah interaksi yang terjadi pada *powder*, atau dengan kata lain tumbukan antara *powder* dengan *powder* lain, dengan *ball*, dan juga dengan dinding *vial*. Adapun persyaratan dari tumbukan-tumbukan tersebut adalah sebagai berikut [13]:

- Tumbukan antara *powder* dengan dinding sisi *vial*; tumbukan ini terjadi jika $\left|\vec{\lambda}\right| \geq R_v - R_p$, dengan jari-jari powder R_p, jari-jari *vial* R_v, dan jarak antara titik

pusat koordinat non-inersial pada sistem dengan posisi *powder* $|\vec{\lambda}|$ yang dihitung sebagai $|\vec{\lambda}| \geq y^2+z^2$ dengan *y* dan *z* adalah koordinat *powder* pada sumbu *y* dan *z* sistem non-inersial.

- Tumbukan antara *powder* dengan dasar atau atap *vial*; akan terjadi jika $|x| \geq |L_v/2 - R_p|$ dengan tinggi *vial* L_v dan *x* adalah koordinat *powder* pada sumbu *x* sistem non-inersial.
- Tumbukan antara *powder* dengan *powder* lain dan ball; terjadi jika perpindahan $\xi_{mm'}=0$ atau dengan kata lain $R_m+R_{m'} > |\vec{r}_m - \vec{r}_{m'}|$ dengan R_m jari-jari material, $\vec{r}_m$ posisi material, dan indeks *m* serta *m'* mewakili indeks *b* untuk *ball* dan *p* untuk *powder*.

Kegiatan eksplorasi tetap dilakukan saat melangkah ke tahap selanjutnya yaitu pembuatan program simulasi. Hal tersebut dilakukan karena selama pembuatan program diperlukan evaluasi terus menerus untuk mengatasi bug, meningkatkan efisiensi penggunaan memori, dan peningkatan serta perbaikan lainnya.

4.2.2. Pembuatan Program Simulasi

Berdasarkan hasil eksplorasi, dilakukanlah penulisan program simulasi. Program ditulis dengan program Gedit dan interpreter Python 2.6 pada lingkungan sistem GNU/Linux Ubuntu 9.04. Pembuatan program dilakukan dengan menulis prototipe algoritma program keseluruhan kemudian memperbaiki dan menambahkan tiap detailnya. Kesalahan pemahaman terhadap model matematis seringkali menimbulkan kekeliruan algoritma sehingga penulis perlu melakukan perbaikan secara masif. Secara umum proses pembuatan program dilakukan dengan metode trial dan error.

Penulis membagi pembuatan program simulasi ini ke dalam tiga bagian: pembuatan algoritma *pseudo-random number generator*, program utama, dan *parallel computing*.

4.2.2.1. Pembuatan *Pseudo-Random Number Generator*

Untuk keperluan simulasi dengan Monte Carlo, algoritma *pseudo-random* LCG saja tidak cukup. Karena itu, penulis mempelajari beberapa algoritma *Pseudo-random* dan memilih untuk menggunakan algoritma Wichmann-Hill. Algoritma *pseudo-random* ini cukup mudah untuk ditulis karena hanya membutuhkan algoritma yang pendek serta memiliki periode yang cukup besar dan mencukupi kebutuhan pada simulasi yang akan dilakukan yaitu sekitar 2^{120} [19]. Salah satu kekurangannya adalah lama waktu pembangkitan yang tergolong lama dibandingkan algoritma lainnya. Sebagai gambaran pembanding, algoritma Mersenne-Twister yang digunakan sebagai algoritma *pseudo-random* standar pada Python memiliki periode sekitar 2^{19937}-1 dan waktu pembangkitan yang tergolong cepat. Namun algoritma Wichmann-Hill lebih mudah ditulis, karenanya algoritma tersebut dipilih. Penulis membangun algoritma tersebut berdasarkan referensi [18] dan [19].

Awalnya penulis mencoba untuk membuat *pseudo-random number generator* yang memiliki fungsi pembatasan jumlah angka desimal. Namun hal ini sangat sulit dilakukan karena algoritma Wichmann-Hill tidak memungkinkan hal tersebut. Selain itu, fungsi pembulatan angka pada Python yaitu round( ) tidak membulatkan angka sesuai dengan yang diharapkan. Sebagai ilustrasi, round(2.4563429,4) akan menghasilkan keluaran 2.4563000000000001 padahal seharusnya 2.4563 saja. Hal ini dapat diatasi dengan membuat fungsi pembulatan yang berdasarkan tipe data ‘decimal’ yang merupakan modul pada Python untuk melakukan operasi pada bilangan desimal sesuai dengan yang diharapkan. Meskipun begitu, hal ini tidak banyak membantu.

4.2.2.2. Pembuatan Program Utama

Program utama ditulis dengan mengikuti alur perhitungan pada model matematis. Persoalan yang sangat sulit diatasi adalah keterbatasan kemampuan komputer yang digunakan untuk menjalankan program simulasi yang memang diharuskan

untuk dapat melakukan komputasi dan iterasi dalam jumlah yang besar. Seringkali program mengalami *crash* saat menjalankan program bahkan hingga mengakibatkan sistem operasi *hang*. Hal ini tidak dapat diatasi bahkan dengan menggunakan komputer lain sedangkan pada saat itu *cluster-cluster* komputer belum siap untuk digunakan secara parallel. Hal ini terjadi karena penggunaan algoritma yang kurang efisien terhadap memori. Untuk mengatasi hal tersebut, dilakukan hal-hal berikut:

1. Mengganti fungsi range( ) pada algoritma pengulangan 'for ... in ...' dengan xrange( ) yang lebih ramah memori. Namun fungsi xrange( ) masih memiliki keterbatasan yaitu hanya bisa digunakan hingga nilai input fungsi yang terbatas (tergantung besar memori komputer yang digunakan). Karenanya penulis membuat algoritma fungsi generator berdasarkan algoritma yang ditulis oleh Danny Yoo pada [17]. Dengan ini r*unning time* program menjadi sedikit lebih lama namun iterasi dalam jumlah besar menjadi lebih mungkin untuk dilakukan.
2. Menuliskan hasil penghitungan pada file, hal ini dapat memperingan beban memori meski *running time* menjadi sedikit lebih lambat. Ruang *harddisk* yang cukup harus disediakan untuk menampung file-file keluaran yang dihasilkan oleh program.
3. Tidak menampilkan hasil perhitungan pada layar, hal ini mempercepat *running time* dan sedikit meringankan memori.
4. Tidak melakukan visualisasi grafis. Visualisasi grafis menggunakan modul vpython menggunakan cukup banyak sumber daya memori, terlebih untuk jumlah iterasi yang sangat besar. Visualisasi juga tidak begitu berarti pada jumlah iterasi yang sangat besar karena visualisasi yang ditampilkan akan saling tumpang tindih.

Namun, penulis pada akhirnya tidak dapat mereproduksi program simulasi seperti yang sudah dibuat sebelumnya. Program simulasi yang dibuat penulis masih menghasilkan keluaran yang berbeda untuk input yang sama. Setelah dipelajari lebih lanjut, algoritma perhitungan yang tertulis pada program simulasi tidak

langsung mengadopsi prosedur perhitungan pada persamaan (4.10) sesuai dengan ekspektasi penulis.

Beberapa hal pada program simulasi semula yang tidak sesuai dengan ekspektasi penulis dapat diketahui logikanya, sedangkan beberapa hal lainnya tidak dapat diketahui. Hal-hal yang di luar ekspektasi tersebut adalah:

- Sistem koordinat dari spex mill yang digunakan sebagai model ball mill menentukan variabel integrasi dan batas integrasinya. Sistem koordinat silinder lebih realistis untuk diterapkan mengingat bentuk *vial* spex mill yang berupa silinder. Namun sistem koordinat tersebut mengakibatkan program menghasilkan output yang terlalu besar karena terdapat konstanta hasil transformasi koordinat yang dikalikan berulang-ulang selama iterasi penghitungan. Karenanya digunakan sistem koordinat non-inersial yang berbasis Cartesian.

- Pada model matematis yang disimulasikan, tumbukan yang diperhitungkan adalah antara partikel bubuk (*powder*) dengan *ball*, *powder* dengan *powder* lainnya, dan *powder* dengan dinding *vial*. Berlainan dengan hal tersebut, program simulasi awal tidak menghitung tumbukan antara *powder* dengan dinding *vial*.

- Program simulasi awal menjumlahkan nilai variable *displacement* ($\xi_{mm'}$) pada setiap kejadian tumbukan kemudian menghitung pangkatnya dengan 5/2 (2,5). Hal ini berbeda dengan model matematis yang melakukan penghitungan variabel $\xi_{mm'}$ pangkat 5/2 sebelum melakukan operasi *summation* (penjumlahan) terhadap nilai variabel $\xi_{mm'}$ untuk setiap kejadian tumbukan dalam satu perulangan pada integrasi Monte Carlo.

- Pada saat menghitung fungsi akhir, nilai variable *displacement* ($\xi_{mm'}$) dirata-ratakan sesuai dengan jumlah masing-masing jenis tumbukan yang terjadi. Padahal, hal seperti ini tidak ditunjukkan pada model matematis.

- Jari-jari efektif ($R_{mm'}^{eff}$) dijumlahkan setiap kejadian tumbukan, kemudian baru dihitung akar kuadratnya. Pada model matematis yang diberikan, ditunjukkan bahwa akar dari radius efektif memiliki nilai yang konstan dan dijumlahkan setelah dihitung akar kuadratnya.

- Syarat tumbukan pada kasus tumbukan *ball* dan *powder* bias. Jumlah tumbukan dihitung jika $R_{\mathrm{p}}+R_{\mathrm{b}} > \left|\vec{r}_p - \vec{r}_b\right|$, sedangkan total *displacement* (ξ_{pb}) dihitung dengan syarat tambahan yaitu $\xi_{pb} > R_p$. Syarat tambahan tersebut ditambahkan sebab tidak mungkin terdapat kejadian *powder* masuk ke dalam *ball* yang akan menjadikan *ball* suatu komposit. Namun penulis tidak mengetahui mengenai alasan jumlah tumbukan yang langsung dihitung tanpa perlu melalui syarat tambahan.

- Penghitungan nilai akhir pada program simulasi sangat berbeda dengan fungsi $\mathcal{F}$ yang tertulis pada paper [1], [2], maupun [3]. Salah satu yang mencolok adalah suku kedua dan ketiga pada fungsi $\mathcal{F}$ ditulis positif, padahal pada model matematis yang diberikan, kedua suku tersebut memiliki tanda negatif. Selain itu, angka 2 pada model matematis dikalikan dengan seluruh fungsi $\mathcal{F}$, sedangkan pada program simulasi awal angka 2 hanya dikalikan pada suku pertama dari fungsi $\mathcal{F}$.

Penulis tetap mengikuti logika pada program simulasi awal agar memiliki referensi untuk hasil simulasi. Untuk alasan yang sama, penulis tidak dapat mencoba memasukkan variabel waktu pada program simulasi. Pengujian simulasi program utama dilakukan dengan memberikan masukkan sebagai berikut:

- Material *powder* yang disimulasikan adalah nikel dengan modulus Young sebesar 207 Gpa dan rasio Poisson sebesar 0.31,
- *Spex mill* memiliki spesifikasi sebagai berikut: panjang lengan mekanik (L) 200 mm, panjang *vial* (L_v) 50 mm, jari-jari *vial* (R_v), jari-jari *ball* (R_b) 5 mm,
- Jari-jari *powder* (R_p) 0.3 mm sehingga perbandingan R_b : R_p sebesar 50:3,

- Perbandingan jumlah *powder* (n_p) : jumlah *ball* (n_b) adalah 200:1,
- Simulasi dilakukan pada kondisi sistem statis pada satu waktu tertentu sehingga sudut α dan θ konstan pada nilai tertentu yang pada simulasi ini diberikan nilai 1^o,
- Parameter medan elektromagnetik tidak dilibatkan.

4.2.2.3. Pembuatan Algoritma *Parallel Computing*

Sebagaimana telah dijelaskan sebelumnya, untuk menggunakan algoritma komputasi paralel pada pemrograman Python dengan interface OpenMPI, penulis menggunakan perangkat lunak PyMPI. Fungsi PyMPI adalah menjembatani antara Python dengan fungsi-fungsi yang ada pada OpenMPI. PyMPI menghasilkan modul mpi yang dapat digunakan langsung dengan perintah import.

Pada program simulasi yang penulis buat, algoritma komputasi paralel ditekankan implementasinya pada algoritma *pseudo-random number generator* karena pembangkitan bilangan acak merupakan salah satu proses yang paling banyak dilakukan pada program utama. Kebutuhan utamanya adalah membuat algoritma *pseudo-random number generator* yang dapat membangkitkan nilai acak yang berbeda-beda pada tiap proses paralel.

Algoritma komputasi paralel juga diimplementasikan pada program utama khususnya dalam penghitungan nilai akhir dan algoritma untuk menuliskan hasil perhitungan ke dalam file-file yang terpisah. Komputasi parallel dapat diaplikasikan lebih luas lagi dalam program simulasi, namun dengan implementasi yang sedikit seperti yang sudah dilakukan pun sudah cukup mempersingkat *running time*.

4.2.3. Pelaporan

Proses pelaporan hasil kerja praktek dilakukan pada tahap akhir kerja praktek di PPF LIPI. Pelaporan hasil kerja praktek ini dilakukan dengan pembuatan laporan kerja praktek.

4.3. Pencapaian Hasil

Kerja praktek yang penulis laksanakan ini menghasilkan program simulasi baru untuk pemodelan dinamika internal pada proses penumbukan dalam *ball mill* sebagai *canonical ensemble* dengan menggunakan metode integrasi Monte Carlo pada bahasa pemrograman Python. Program simulasi yang dibuat tidak memenuhi seluruh spesifikasi yang diberikan pada deskripsi pekerjaan. Adapun spesifikasi yang berhasil dicapai adalah:

- Menghasilkan bilangan random dengan algoritma buatan sendiri (bukan bawaan Python) dan dapat digunakan pada sistem komputasi paralel. Akan tetapi, *pseudo-random number generator* ini tidak dapat membangkitkan bilangan acak dengan jumlah angka desimal yang dibatasi.
- Melakukan integrasi Monte Carlo untuk menghasilkan nilai temperatur yang merepresentasikan evolusi jumlah partikel.
- Dapat menghasilkan nilai keluaran simulasi yang sesuai dengan program simulasi sebelumnya.
- Mendukung komputasi paralel. Penggunaan komputer paralel pada simulasi ini terbukti dapat mempersingkat *running time* dengan peningkatan yang hampir linear sesuai dengan jumlah komputer paralel yang digunakan.

Program simulasi ini telah diuji untuk berjalan menggunakan 5 inti prosesor pada prosesor dengan 6 inti dan RAM 3,4 GB. Dengan input yang telah dijelaskan sebelumnya yaitu material *powder* berupa nikel, R_p : R_b sebesar 50:3, n_p : n_b sebesar 200:1, dan jumlah iterasi (N) sebanyak 10^5, program simulasi menghasilkan nilai $\mathcal{F}$ sebesar 2,5845905863 x 10^{-14} dalam waktu 21 jam lebih 26

menit. Nilai tersebut kurang lebih berkesuaian dengan nilai $\mathcal{F}$ pada paper [2] untuk input yang sama.

Program simulasi yang dibuat oleh penulis memiliki beberapa perbedaan dengan program simulasi yang dibuat sebelumnya. Sebagian menunjukkan peningkatan kinerja sedangkan sebagian lainnya tidak terlalu berpengaruh. Perbedaan-perbedaan tersebut antara lain:

- Tidak melakukan perintah visualisasi melalui modul vpython. Alasannya sudah dijelaskan pada sub-bab 4.2.2.2.
- Tidak mencetak hasil pada layar, melainkan menuliskan keluaran-keluaran pada file keluaran seperti yang sudah dijelaskan sebelumnya pada sub-bab 4.2.2.2. Hal ini juga dilakukan karena pemantauan hasil perhitungan secara terus menerus dirasa tidak diperlukan.
- Untuk menghasilkan bilangan random digunakan fungsi random yang dibuat sendiri berdasarkan algoritma *pseudo-random number generator* Wichmann-Hil. Sub-rutin fungsi random diletakkan dalam file terpisah dengan file utama.
- Fungsi transformasi koordinat serta variabel-variabel yang memuat karakteristik ball mill diletakkan dalam file yang terpisah. Hal ini dimaksudkan untuk memudahkan pengubahan fungsi jika model ball mill yang digunakan dalam simulasi akan diganti.
- Parameter-parameter masukan simulasi ditulis dalam file terpisah. Hal ini dimaksudkan agar tidak perlu mengisi input satu-satu setiap kali simulasi sehingga menyulitkan terutama saat melakukan pengujian program dengan input yang sama berkali-kali.
- Algoritma, penamaan variabel, serta struktur program sedikit berbeda. Perubahan yang dilakukan umumnya untuk menyederhanakan perhitungan dan menghapus sintaks-sintaks yang tidak terlalu diperlukan. Meskipun demikian, keluaran yang dihasilkan sama dengan program sebelumnya.

- Pada setiap file terdapat komentar-komentar yang membantu menjelaskan isi dari setiap file tersebut.
- Total ukuran program simulasi lebih besar dari sebelumnya.
- Program sebelumnya menggunakan satuan milimeter pada input sehingga pada perhitungan harus mengkonversinya menjadi satuan meter. Program baru menggunakan satuan meter pada input.
- Waktu simulasi relatif lebih cepat. Sebagai perbandingan, pada jumlah iterasi 10 (N=10), program sebelumnya membutuhkan waktu 105 detik sedangkan program buatan penulis membutuhkan waktu 87 detik (lebih cepat ~18%).
- Program simulasi baru sudah dapat digunakan pada komputasi paralel.

Untuk memudahkan pemanggilan fungsi dan pengubahannya di kemudian hari, program dipecah menjadi beberapa file. File-file tersebut adalah:

- millinput.py; berisi variabel-variabel input yang meliputi karakter material, jari-jari *powder* dan *ball*, serta jumlah iterasi yang diinginkan.
- generator.py; berisi fungsi-fungsi pembantu simulasi seperti *pseudo-random number generator* yang tersedia dalam berbagai metode algoritma dan fungsi generator untuk digunakan pada proses iterasi menggantikan fungsi xrange( ).
- vial.py; berisi fungsi yang berhubungan dengan geometri ball mill seperti fungsi transformasi koordinat non-inersial ke koordinat inersial dan fungsi penghitungan konstanta transformasi koordinat.
- main.py; berisi program utama yang memuat algoritma integrasi Monte Carlo, algoritma penulisan keluaran pada file, dan penghitungan hasil akhir dari simulasi model matematis.

Program simulasi yang telah dibuat memang masih jauh dari harapan. Akan tetapi, hal-hal yang telah dicapai dapat membuka jalan untuk pengembangan program simulasi lebih lanjut.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kesimpulan dari hasil kerja praktek ini adalah:

- Komputasi fisika dalam sains dan teknologi material merupakan hal yang penting untuk dilakukan karena dapat membantu untuk mempemudah untuk mempelajari perilaku material dalam suatu proses tertentu.
- Bahasa pemrograman Python merupakan salah satu bahasa pemrograman yang sesuai untuk keperluan komputasi sains.
- Program simulasi yang dibuat menghasilkan keluaran yang sesuai dengan penelitian sebelumnya namun memiliki kinerja yang lebih optimal dalam kecepatan dengan *running time* yang lebih cepat 18% dari sebelumnya.
- *Parallel computing* dengan interface OpenMPI dapat membantu program simulasi agar bekerja lebih cepat.

5.2. Saran

Saran-saran yang dapat penulis sampaikan untuk pengembangan penelitian selanjutnya adalah:

- Menguji program simulasi pada resolusi yang diharapkan yaitu orde puluhan mikrometer atau sekitar ~ 10^8 x 10^8 x 10^8.
- Menggunakan metode Monte Carlo yang lebih mutakhir untuk kinerja simulasi dan hasil simulasi yang lebih baik.
- Menggunakan *pseudo-random number generator* bawaan Python yaitu yang berbadasarkan algoritma Mersenne-Twister karena algoritma tersebut dibuat untuk kebutuhan analisis numerik dengan Monte Carlo dan mendukung

penggunaan komputasi paralel. Akan lebih baik jika algoritma tersebut dapat dimodifikasi sehingga dapat membatasi jumlah angka desimal dari bilangan yang dibangkitkan olehnya.

- Memasukkan parameter waktu pada simulasi, atau dengan kata lain, memasukkan unsur perubahan sudut α dan θ pada simulasi.
- Mencoba menggunakan interface komputasi paralel selain openMPI untuk membandingkan interface yang paling baik untuk kebutuhan simulasi.

DAFTAR PUSTAKA

[1] Muhandis, F. Nurdiana, A.S.Wismogroho, N.T. Rochman, dan L.T. Handoko, "Extracting Physical Observables Using Macroscopic Ensemble in The Spex-Mixer/Mill Simulation," AIP Proceeding Supplement, vol. 1169, pp. 235–240, 2009.

[2] G. K. Sunnardianto, Muhandis, F. N. Diana, dan L.T. Handoko, "Modeling Comminution Processes in Ball Mills as a Canonical Ensemble," J. Compt. Theor. Nanoscience, vol. 8, pp. 194-200, 2011.

[3] G. K. Sunnardianto dan L.T. Handoko, "Macroscopic Simulation for The Internal Dynamics of Nano Processes in Ball Mills," Proceeding of the 4th International Joint Conference on Integrated Systems, Design and Technology, 2010.

[4] "Renstra LIPI 2010-2014," Lembaga Ilmu Pengetahuan Indonesia, Jakarta, 2010.

[5] "Fisika LIPI", http://fisika.lipi.go.id/.

[6] "Situs Web Kawasan Puspiptek", http://puspiptek.net/.

[7] "Grup Fisika Teoritik dan Komputasi – GFTK", http://teori.fisika.go.id.

[8] "Peta riset GFTK", http://sains.org/handoko/photo/roadmap_riset.jpg.

[9] R. D. Skeel, *Computing for Science and Engineering*. 2008.

[10] J. Kiusaalas, *Numerical Methods in Engineering With* Python. New York: Cambridge University Press, 2005.

[11] S. Weinzierl, *Introduction to Monte Carlo Methods*. Amsterdam: NIKHEF Theory Group, 2000.

[12] S. Moody dan M. Perl, "Parallel Computing 101", Utah State University, Utah, 2007.

[13] A. Concas, N. Lai, M. Pisu, dan G. Cao, "Modelling of comminution Processes in Spex Mixer/Mill," Chemical Engineering Science, vol. 61, pp. 3746–3760, 2006.

[14] P. Miller, “Parallel, Distributed Scripting with Python”, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, University of California, 2002.

[15] P. Miller, “pyMPI – An Introduction to Parallel Python Using MPI”, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, University of California, September 2002.

[16] C. Hoffman, R. Gruet, et. al. , “Python 2.6 Quick Reference”, Februari 2009, http://rgruet.free.fr/PQR26/PQR2.6_modern_a4.pdf.

[17] D. Yoo (dyoo@hkn.eecs.berkeley.edu), "Re: [Tutor] Workaround for limitation in xrange()?," mailing list post to Tutor@python.org mailing list, 10 Oktober, 2006.

[18] anhn, "Wichmann-Hill Pseudo Random Number Generator: an alternative for VB Rnd() function," post to vbforums.com, 5 Desember, 2007.

[19] B. A. Wichmann dan I. D. Hill, “Generating Good Pseudo-Random Numbers,” UK, 5 Desember 2005. diunduh dari: http://www.eurometros.org/file_download.php?file_key=247.