TUGAS AKHIR SM-141501

DEKOMPOSISI DATA DALAM SIMULASI MODEL PENYEBARAN ALIRAN DEBRIS SATU DAN DUA DIMENSI

Mukhamad Wiwid Setiawan
NRP 1211 100 022

Dosen Pembimbing
Drs. Soetrisno, MI.Komp.
Dr. Imam Mukhlash S.Si, MT

JURUSAN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT SM-141501

DATA DECOMPOSITION IN SIMULATION OF ONE AND TWO DIMENSIONAL DEBRIS FLOW DISTRIBUTION MODEL

Mukhamad Wiwid Setiawan
NRP 1211 100 022

Supervisor
Drs. Soetrisno, MI.Komp.
Dr. Imam Mukhlash S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty of Mathematics and Natural Science
Sepuluh Nopember Institute of Technology
Surabaya 2015

LEMBAR PENGESAHAN

DEKOMPOSISI DATA DALAM SIMULASI MODEL PENYEBARAN ALIRAN DEBRIS SATU DAN DUA DIMENSI

DATA DECOMPOSITION IN SIMULATION OF ONE AND TWO DIMENSIONAL DEBRIS FLOW DISTRIBUTION MODEL

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Sains
Pada Bidang Studi Ilmu Komputer
Program Studi S-1 Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
MUKHAMAD WIWID SETIAWAN
NRP. 1211 100 022

Menyetujui,

Dosen Pembimbing II,	Dosen Pembimbing I,
Dr. Imam Mukhlash, S.Si, MT	Drs. Soetrisno, MI.Komp.
NIP. 19700831 199403 1 003	NIP. 19571103 198603 1 003

Mengetahui,
Ketua Jurusan Matematika
FMIPA-ITS

Prof. Dr. Erna Apriliani, M.Si
NIP. 19660414 199102 2 001

Surabaya, Juli 2015

DEKOMPOSISI DATA DALAM SIMULASI MODEL PENYEBARAN ALIRAN DEBRIS SATU DAN DUA DIMENSI

NamaMahasiswa	**: Mukhamad Wiwid Setiawan**
NRP	**: 1211 100 022**
Jurusan	**: Matematika**
DosenPembimbing	**: Drs. Soetrisno, MI. Komp.**
	Dr. Imam Mukhlash S.Si, MT.

Abstrak

Penelitian ini bertujuan untuk mengkaji dekomposisi data DEM-SRTM agar data ini dapat diproses dalam *software* simulasi aliran debris dengan hasil *running time* yang lebih rendah dan akurasi hasil simulasi tetap dipertahankan. Tahapan dalam simulasi aliran debris satu dan dua dimensi pada Tugas Akhir ini meliputi tahap pengolahan DEM, pengolahan data simulasi dan simulasi aliran debris. Proses pengolahan DEM adalah proses penarikan data ketinggian satu dan dua dimensi dari DEM. Proses pengolahan data simulasi merupakan proses untuk mendapatkan data-data untuk simulasi. Proses Simulasi aliran debris adalah proses melakukan simulasi aliran debris dengan menggunakan dekomposisi data. Berdasarkan hasil pengujian, teknik dekomposisi data berhasil meningkatkan kecepatan komputasi dalam simulasi aliran debris satu dan dua dimensi sebesar 41.5769%.

KataKunci: DEM, Dekomposisi Data, Simulasi Aliran Debris.

“Halaman ini sengaja dikosongkan”

DATA DECOMPOSITION IN SIMULATION OF ONE AND TWO DIMENSIONAL DEBRIS FLOW DISTRIBUTION MODEL

Name	***: Mukhamad Wiwid Setiawan***
NRP	***: 1211 100 022***
Department	***: Mathematics***
Supervisor	***: Drs. Soetrisno, MI. Komp.***
	Dr. Imam MukhlashS.Si, MT.

Abstract

This study aims to assess DEM-SRTM data decomposition so this data can be processed in a debris flow simulation software which has a lower running time result and accuracy of simulation results is maintained. Stages in simulation of one and two dimensional debris flow in this final project includes DEM processing, data processing and debris flow simulation step. DEM processing is a process of retrieving the height data of one and two dimensional from DEM. Data simulation processing is a process to obtain data for simulation. Debris flow simulation process is a process to simulate debris flow using data decomposition. Based on test results, data decomposition technique succeeded in improving computing speed in a one and two dimensional debris flow simulation of 41.5769%.

Keywords: DEM, Data Decomposition, Debris Flow Simulation.

“Halaman ini sengaja dikosongkan”

KATA PENGANTAR

Segala Puji bagi Allah SWT Tuhan semesta alam yang telah memberikan karunia, rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul: **"Dekomposisi Data Dalam Simulasi Model Penyebaran Aliran Debris Satu Dan Dua Dimensi"** yang merupakan salah satu persyaratan akademis dalam menyelesaikan Program Studi S-1 pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan berkat kerjasama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis mengucapkan terima kasih kepada:

1. Drs. Soetrisno, MI.Komp. dan Dr. Imam Mukhlash, S.Si, MT selaku dosen pembimbing yang senantiasa membimbing dengan sabar dan memberikan kritik dan saran dalam penyusunan Tugas Akhir ini.
2. Prof. Dr. Erna Apriliani, M.Si selaku Ketua Jurusan Matematika.
3. Dra. Sri Suprapti H., M.Si selaku Dosen Wali.
4. Dr. Dwi Ratna Sulistyaningrum, S.Si, MT, Dr. Chairul Imron, MI.Komp., dan Kistosil Fahim, M.Si. selaku dosen penguji Tugas Akhir ini.
5. Dr. Chairul Imron, MI.Komp. selaku Koordinator Program Studi S1 Jurusan Matematika ITS.
6. Seluruh jajaran dosen dan staf jurusan Matematika ITS.
7. Teman-teman mahasiswa jurusan Matematika ITS.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan saran dan kritik dari pembaca. Akhir kata, semoga Tugas Akhir ini bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, Juli 2015

Penulis

special thanks to

Selama proses pembuatan Tugas Akhir ini, banyak pihak yang telah memberikan bantuan dan dukungan untuk penulis. Penulis mengucapkan terima kasih dan apresisasi secara khusus kepada:

1. Ayah Mukhamad Kusaeri dan Ibu Erni Samiatin yang telah senantiasa dengan ikhlash memberikan bimbingan, semangat, nasehat, kasih sayang, perhatian, doa, dan motivasi yang sangat berarti bagi penulis.
2. Adik Reni Dwi Setyowati yang telah memberikan dorongan yang besar bagi penulis agar dapat segera menyelesaikan Tugas Akhir dengan baik.
3. Achmad Fatoni, Liyana, Hakam, Handy, Mas Ridho, Mas Romi dan kawan-kawan pejuang wisuda 112 yang telah memberi motivasi, semangat dan saran agar penulis dapat segera lulus.
4. Rifdy, Selvi, Syukron, Angga dan teman-teman angkatan 2011 yang tidak bisa disebutkan satu per satu, terimakasih atas segala bentuk semangat dan dukungannya kepada penulis.

Tentu saja masih banyak pihak lain yang turut andil dalam penyelesaian tugas akhir ini yang tidak bisa penulis sebutkan satu per satu. Semoga Allah membalas dengan balasan yang lebih baik bagi semua pihak yang telah membantu penulis. *Amin yarabbal 'alamin*.

DAFTAR SIMBOL

Simbol	Keterangan
h	Kedalaman aliran
v	Kecepatan
M	Mass flux ($M = vh$)
g	Percepatan gravitasi
Z_b	Ketinggian lereng
H	Hasil penjumlahan dari kedalaman aliran dan ketinggian lereng ($H = h + Z_b$)
c	Konsentrasi sedimen
ρ_m	Massa jenis aliran ($\rho_m = (\sigma - \rho)c + \rho$)
ρ	Massa jenis air (1 g/cm^3)
σ	Massa jenis sedimen (2.65 g/cm^3)
S_T	Kecepatan erosi
τ_b	Tegangan gaya
τ_{bx}	Tegangan gaya pada sumbu-*x*
τ_{by}	Tegangan gaya pada sumbu-*y*
σ_b	Tegangan geser
u	Kecepatan aliran pada sumbu-*x*
v	Kecepatan aliran pada sumbu-y
M	Mass flux pada sumbu-*x*
N	Mass flux pada sumbu-*y*
t	Waktu
x	Jarak
Ld	Jarak antara titik pusat dam dan puncak gunung
$B1$	Jarak antara titik pusat dam dan titik awal dam
$B2$	Jarak antara titik pusat dam dan titik akhir dam
Hd	Ketinggian dam
A	Luas sungai
Ls	Panjang sungai
Ht	Beda tinggi hulu hilir
$R24$	Curah hujan harian
f	Koefisien pengaliran

“Halaman ini sengaja dikosongkan”

DAFTAR ISI

DAFTAR TABEL

“Halaman ini sengaja dikosongkan”

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Bab ini membahas latar belakang permasalahan yang mendasari penulisan Tugas Akhir ini, kemudian inti dari masalah tersebut disusun dalam bentuk rumusan masalah disertai dengan batasan masalah untuk membatasi pembahasan. Terdapat pula tujuan dan manfaat serta sistematika penulisan Tugas Akhir ini.

1.1 Latar Belakang

Aliran debris adalah aliran dengan konsentrasi sedimen tinggi pada sungai dengan kemiringan curam. Aliran debris meluncur dengan kecepatan tinggi dan memiliki daya rusak besar sehingga mengancam kehidupan manusia, menimbulkan kerugian harta dan benda serta kerusakan lingkungan[8].

Kali Putih merupakan daerah bahaya Gunung Merapi tipe 1 yang sering mengalami bencana aliran debris. Aliran debris yang dapat di tampung pada daerah tipe 1, 2, dan 3 sejumlah 56.602 x 10^3 m^3 sedangkan letusan Gunung Merapi sampai saat ini di prediksi mencapai 200 juta m^3 sehingga apabila turun hujan didaerah Gunung Merapi maka akan terjadi *overload*[3]. Dengan memprediksi distribusi aliran debris, maka dapat diperoleh informasi penyebaran aliran debris yang dapat digunakan untuk mitigasi bencana aliran debris khususnya di Kali Putih.

Sebelumnya telah terdapat penelitian oleh Bandung Arry Sanjoyo dan Dieky Adzkiya, berkaitan dengan hal tersebut yang berjudul "Simulasi Penyebaran Aliran Debris Satu dan Dua Dimensi Menggunakan Metode Beda Hingga". Pada penelitian tersebut, diberikan diskritisasi model penyebaran aliran debris satu dan dua dimensi dengan menggunakan skema leap-frog. Hasil diskritisasi tersebut digunakan untuk membangun *software* yang mampu mensimulasikan aliran debris di lereng gunung[4].

Data ketinggian adalah salah satu *input* utama pada *software* ini tetapi data tersebut memiliki resolusi spasial yang dianggap terlalu besar. Resolusi spasial adalah luas suatu objek di bumi

yang diukur dalam satuan piksel pada citra satelit. Semakin besar resolusi spasial data tersebut maka semakin rendah tingkat kedetailan objek yang terekam[9]. Oleh karena itu, data ketinggian tersebut perlu diganti dengan data ketinggian lain yang memiliki resolusi spasial lebih kecil (tingkat kedetailan lebih besar) untuk meningkatkan akurasi hasil simulasi.

Data DEM telah digunakan sebagai salah satu data utama untuk mendukung kegiatan pembuatan peta topografi, koreksi citra satelit, pemetaan daerah rawan bencana (banjir, tsunami, longsor, dan gunung api) serta penyusunan tataruang wilayah. Kelebihan pada data DEM SRTM adalah tingkat akurasi yang tinggi namun dengan resolusi spasial yang rendah apabila dibandingkan dengan DEM lain seperti DEM ALOS PRIS dan DEM dari peta topografi. Sehingga DEM SRTM sesuai untuk digunakan sebagai data pengganti pada simulasi aliran debris karena tingkat kedetailannya yang lebih besar, selain itu data ini lebih mudah diperoleh [7].

Namun data yang diolah sekarang menjadi sangat besar dan kompleks melebihi ukuran data yang mampu diproses komputer, sehingga CPU tidak mampu merespon (*error*) saat menjalankan simulasi. Solusi pada permasalahan ini adalah dengan menambahkan teknik dekomposisi untuk membagi data tersebut menjadi lebih kecil, minimal sebesar data maksimum yang dapat dijalankan dalam software simulasi. Teknik dekomposisi yang akan digunakan adalah dekomposisi data *input* yaitu membagi data *input* menjadi lebih kecil agar mampu untuk dijalankan dengan setiap sub-data terhubung. Teknik ini tidak merubah isi dan karakter data karena teknik ini merupakan teknik pembacaan data. Pada akhir dari setiap proses perhitungan terhadap sebuah subdata, akan diperoleh data tambahan yang digunakan untuk mengurangi jumlah akses terhadap data selanjutnya. Simulasi kemudian dijalankan hingga semua data yang dibutuhkan terakses. Hasil simulasi tersebut kemudian digabungkan dengan memperhatikan hubungan antar subdata.

1.2 Rumusan Masalah

Berdasarkan latar belakang maka rumusan masalah dalam Tugas Akhir ini adalah bagaimana mengkaji dekomposisi data pada data DEM-SRTM, agar data ini dapat diproses dalam software simulasi aliran debris dengan hasil running time yang lebih rendah, dan akurasi hasil simulasi tetap dipertahankan.

1.3 Batasan Masalah

Batasan masalah dalam Tugas Akhir ini adalah sebagai berikut:

1. Studi ini dikhususkan pada daerah Kali Putih lereng Merapi.
2. Model aliran debris satu dan dua dimensi yang digunakan adalah persamaan-persamaan yang diperoleh dengan menggunakan metode beda hingga.
3. Simulasi model penyebaran aliran debris menggunakan perangkat lunak MATLAB.
4. *Software* simulasi menggunakan data DEM-SRTM dengan resolusi spasial 90 meter yang diperoleh dari BPPTK.

1.4 Tujuan

Tujuan yang ingin dicapai dalam Tugas akhir ini adalah mengkaji dekomposisi data pada data DEM-SRTM, agar data ini dapat diproses dalam software simulasi aliran debris dengan hasil running time yang lebih rendah dan akurasi hasil simulasi tetap dipertahankan.

1.5 Manfaat

Manfaat yang dapat diperoleh dari Tugas Akhir ini adalah sebagai berikut:

1. Dapat menyelesaikan permasalahan penggunaan data yang besar.
2. Dapat meningkatkan akurasi hasil simulasi.
3. Meningkatkan kualitas pengambilan keputusan untuk mitigasi bencana aliran debris.

1.6 Sistematika Penulisan

Sistematika penulisan didalam Tugas Akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini berisi tentang gambaran umum dari penulisan Tugas Akhir ini yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan dan manfaat serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini berisi tentang materi-materi yang mendukung Tugas Akhir ini, antara lain model aliran debris satu dan dua dimensi, perhitungan debit, data DEM, dan teknik dekomposisi.

BAB III METODOLOGI PENELITIAN

Pada bab ini dibahas tentang metode yang digunakan untuk menyelesaikan Tugas Akhir.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dibahas mengenai analisis yang meliputi analisis pengolahan data DEM dan analisis dekomposisi data, perancangan data, perancangan algoritma yang meliputi perancangan pengolahan data DEM, perancangan pengolahan data simulasi dan perancangan simulasi aliran, dan hasil implementasi yang meliputi implementasi antarmuka, implementasi pengolahan data DEM, implementasi pengolahan data simulasi dan implementasi simulasi aliran.

BAB V PENGUJIAN DAN PEMBAHASAN HASIL

Pada bab ini menyajikan proses pengujian terhadap program. Kemudian dicatat untuk merumuskan kesimpulan dan saran dari Tugas Akhir.

BAB VI KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan yang diperoleh dari hasil penelitian yang telah dilakukan serta saran yang diberikan untuk pengembangan selanjutnya.

BAB II
TINJAUAN PUSTAKA

Bab ini membahas konsep atau teori mengenai model aliran debris satu dan dua dimensi serta hasil diskritisasinya, perhitungan debit banjir, desain software simulasi, data ketinggian, dan teknik dekomposisi data.

2.1 Model Aliran Debris Satu Dimensi

Penyebaran aliran debris saat menuruni lembah dianggap satu dimensi karena bentuk aliran di lereng cenderung sempit dan dalam sehingga aliran debris diasumsikan mengikuti arah lereng. Penyebaran aliran debris satu dimensi disajikan berupa persamaan differensial parsial dalam empat persamaan, yaitu:

1. Konservasi momentum pada campuran aliran :

$$\frac{\partial M}{\partial t}+\frac{\partial vM}{\partial x}=-gh\frac{\partial H}{\partial x}-\frac{\tau_b}{\rho_m}$$

2. Konservasi massa pada campuran sedimen dan air :

$$\frac{\partial h}{\partial t}+\frac{\partial M}{\partial x}=S_T$$

3. Konservasi massa pada partikel sedimen :

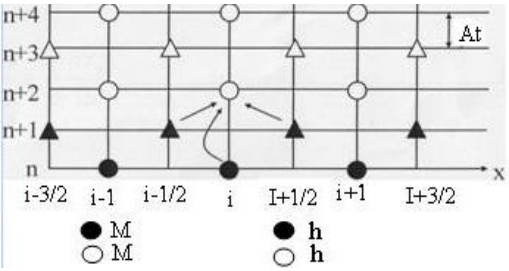
$$\frac{\partial ch}{\partial t}+\frac{\partial cM}{\partial x}=c_*S_T$$

4. Persamaan untuk proses erosi :

$$\mathrm{S_T}=-\frac{\partial \mathrm{Z_b}}{\partial \mathrm{t}}$$

Terdapat dua tahap dalam model ini yaitu metode perhitungan konservasi momentum dan konservasi massa. Metode perhitungan konservasi momentum digunakan untuk menentukan nilai M dan metode perhitungan konservasi massa digunakan untuk menentukan nilai h, c, dan Z_b.

Diskritisasi persamaan tersebut menggunakan metode leapfrog untuk membentuk model dalam bentuk beda hingga, seperti pada Gambar 2.1[5] :



Gambar 2.1 Skema Leapfrog

Hasil diskritisasi dalam model penyebaran aliran debris satu dimensi menggunakan skema leapfrog adalah sebagai berikut[4]:

$$\bar{h} = \frac{h_{i-\frac{1}{2}}^{n+1} + h_{i+\frac{1}{2}}^{n+1}}{2}$$

$$\bar{c} = \frac{c_{i-\frac{1}{2}}^{n+1} + c_{i+\frac{1}{2}}^{n+1}}{2}$$

$$\overline{\rho_m} = (\sigma - \rho)\bar{c} + \rho, \quad \text{dengan } \sigma = 2.65 \text{ dan } \rho = 1$$

$$\theta = arctan\left(\frac{z_b{}_{i+\frac{3}{2}}^{n+1} - z_b{}_{i+\frac{1}{2}}^{n+1}}{\Delta x}\right)$$

$$\sigma_y = (\overline{\rho_m} - \rho)\, g\bar{h}\, cos\theta\, tan\emptyset \left(\frac{\bar{c}}{cst}\right)^{\frac{1}{5}}$$

$$\text{dengan } \emptyset = 38.5° \text{ dan } cst = 0.6$$

$$f = \frac{25}{4}\left\{\frac{k_f(1-\bar{c})^{\frac{5}{3}}}{\bar{c}^{\frac{2}{3}}} + \frac{k_g \sigma}{\rho}(1-e^2)\bar{c}^{\frac{1}{3}}\right\}\left(\frac{\bar{h}}{d}\right)^{-2}$$

$$\text{dengan } k_f = 0.25\,,\ k_g = 0.0828\,, \text{dan } e = 0.85$$

$$v_i = \frac{M_i^n}{\bar{h}}$$

$$v_{i+1} = \frac{2M_{i+1}^n}{\bar{h}_{i+\frac{1}{2}}^{n+1} + \bar{h}_{i+\frac{3}{2}}^{n+1}} \quad \text{sehingga} \quad v_{i+\frac{1}{2}} = \frac{v_{i+1}+v_i}{2}$$

$$v_{i-1} = \frac{2M_{i-1}^n}{\bar{h}_{i-\frac{1}{2}}^{n+1} + \bar{h}_{i-\frac{3}{2}}^{n+1}} \quad \text{sehingga} \quad v_{i-\frac{1}{2}} = \frac{v_{i-1}+v_i}{2}$$

$$XDX = \frac{\overline{v_{i+\frac{1}{2}}}(M_i^n + M_{i+1}^n) + \overline{v_{i-\frac{1}{2}}}(M_{i-1}^n + M_i^n)}{2\Delta x}$$

$$M_i^{n+2} = \frac{-g\bar{h}\frac{h_{i+\frac{1}{2}}^{n+1} + z_{b_{i+\frac{1}{2}}}^{n+1} - h_{i-\frac{1}{2}}^{n+1} - z_{b_{i-\frac{1}{2}}}^{n+1}}{\Delta x} - XDX\frac{\tau_y}{\overline{\rho_m}}\left(\frac{1}{2\Delta t} + \frac{\rho}{\overline{\rho_m}} f \frac{1}{2\bar{h}}\left|\frac{M_i^n}{\bar{h}}\right|\right)}{\frac{1}{2\Delta t} + \frac{\rho}{\overline{\rho_m}} f \frac{1}{2\bar{h}}\left|\frac{M_i^n}{\bar{h}}\right|}$$

$$CXDX = \frac{M_{i+1}^{n+2}\left(c_{i+\frac{1}{2}}^{n+1} - c_{i+\frac{3}{2}}^{n+1}\right) - M_i^{n+2}\left(c_{i-\frac{1}{2}}^{n+1} - c_{i+\frac{1}{2}}^{n+1}\right)}{2\Delta x}$$

$$h_{i+\frac{1}{2}}^{n+3} = \frac{1}{c_* - c_{i+\frac{1}{2}}^{n+3}}\left[h_{i+\frac{1}{2}}^{n+1} - 2\Delta t\left\{c_* \frac{M_{i+1}^{n+2} - M_i^{n+2}}{\Delta x} - CXDX\right\}\right]$$

$$Z_{b_{i+\frac{1}{2}}}^{n+3} = Z_{b_{i+\frac{1}{2}}}^{n+1} + \frac{1}{c_* - c_{i+\frac{1}{2}}^{n+3}}\left[h_{i+\frac{1}{2}}^{n+1} - 2\Delta t\left\{CXDX - c_{i+\frac{1}{2}}^{n+3} \frac{M_{i+1}^{n+2} - M_i^{n+2}}{\Delta x}\right\}\right]$$

$$c_{i+\frac{1}{2}}^{n+3} = \frac{\rho \tan\theta_0}{(\sigma - \rho)(\tan\phi - \tan\phi_0)}$$

2.2 Model Aliran Debris Dua Dimensi

Model aliran debris dua diensi terdiri dari tiga persamaan, yaitu :

1. Persamaan konservasi momentum arah sumbu x :

$$\frac{\partial M}{\partial t}+\frac{\partial}{\partial x}(uM)+\frac{\partial}{\partial y}(vM)=-gh\frac{\partial H}{\partial x}-\frac{\tau_{bx}}{\rho}$$

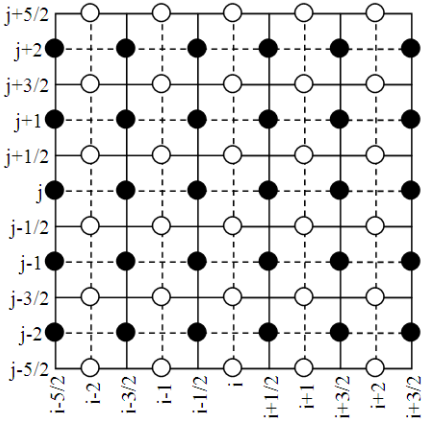
2. Persamaan konservasi momentum arah sumbu y :

$$\frac{\partial N}{\partial t}+\frac{\partial}{\partial x}(uN)+\frac{\partial}{\partial y}(vN)=-gh\frac{\partial H}{\partial x}-\frac{\tau_{by}}{\rho}$$

3. Persamaan kontinuitas :

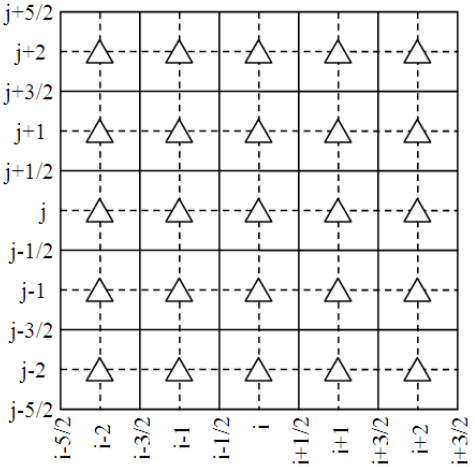
$$\frac{\partial h}{\partial t}+\frac{\partial M}{\partial x}+\frac{\partial N}{\partial y}=0$$

Diskritisasi model sebelumnya dilakukan dengan *staggered scheme*. Metode FTCS *(forward-time, centered-space)* digunakan pada suku konveksi dan *shear stress* untuk menurunkan persamaan pergerakan. Pengaturan dengan skema Leap Frog digunakan dengan persamaan kontinuitas. Posisi diskritisasi untuk *M* dan *N* ditunjukkan oleh Gambar 2.2.



Gambar 2.2 Diskritisasi M dan N

Lingkaran hitam pada Gambar 2.2 menggambarkan diskritissasi untuk $M = u_m h$ sedangkan lingkaran putih menggambarkan diskritisasi posisi untuk $N = v_m h$. Posisi diskritisasi dari variabel h ditunjukkan oleh Gambar 2.3.[1].



Gambar 2.3 Posisi Diskritisasi untuk h

Hasil Diskritisasi dalam model penyebaran aliran debris dua dimensi adalah sebagai berikut [4]:

$$h_{i,j}^{n+1} = 2\Delta t\left(\frac{M_{i+j}^{n} - M_{i,j}^{n}}{\Delta x} + \frac{N_{i,j+1}^{n} - N_{i,j}^{n}}{\Delta y}\right)$$

$$u_{mi+1,j}^{n} = \frac{M_{i+1,j}^{n} + M_{i,j}^{n}}{h_{i+1,j}^{n-1} + h_{i+1,j}^{n+1}} \quad \text{dan} \quad \overline{M_{i+1,j}^{n}} = \frac{M_{i+\frac{1}{2},j}^{n} + M_{i+\frac{3}{2},j}^{n}}{2}$$

$$\frac{\partial(u_m M)}{\partial x} = \frac{u_{mi+1,j}^{n}\,\overline{M_{i+1,j}^{n}} - u_{mi,j}^{n}\,\overline{M_{i,j}^{n}}}{\Delta x}$$

$$v^n_{mi+\frac{1}{2},j+\frac{1}{2}} = \frac{4\left(N^n_{i,j+\frac{1}{2}} + N^n_{i+1,j+\frac{1}{2}}\right)}{h^{n-1}_{i,j} + h^{n-1}_{i,j+1} + h^{n-1}_{i+1,j} + h^{n-1}_{i+1,j+1} + h^{n+1}_{i,j} + h^{n+1}_{i,j+1} + h^{n+1}_{i+1,j} + h^{n+1}_{i+1,j+1}}$$

$$\overline{M^n_{i+\frac{1}{2},j+\frac{1}{2}}} = \frac{M^n_{i+\frac{1}{2},j} + M^n_{i+\frac{1}{2},j+1}}{2}$$

$$\frac{\partial(v_m M)}{\partial y} = \frac{v^n_{mi+\frac{1}{2},j+\frac{1}{2}}\,\overline{M^n_{i+\frac{1}{2},j+\frac{1}{2}}} - v^n_{mi+\frac{1}{2},j-\frac{1}{2}}\,\overline{M^n_{i+\frac{1}{2},j-\frac{1}{2}}}}{\Delta y}$$

$$gh\frac{\partial H}{\partial x} = \frac{h^{n+1}_{i,j} + h^{n+1}_{i+1,j}}{2}\,\frac{Z^{n+1}_{bi+1,j} + h^{n+1}_{i+1,j} - Z^{n+1}_{bi,j} - h^{n+1}_{i,j}}{\Delta x}$$

$$u^n_{mi+\frac{1}{2},j} = \frac{4M^n_{i+\frac{1}{2},j}}{h^{n-1}_{i,j} + h^{n-1}_{i+1,j} + h^{n+1}_{i,j} + h^{n+1}_{i+1,j}}$$

$$v^n_{mi,j+\frac{1}{2}} = \frac{4N^n_{i,j+\frac{1}{2}}}{h^{n-1}_{i,j} + h^{n-1}_{i,j+1} + h^{n+1}_{i,j} + h^{n+1}_{i,j+1}}$$

$$v^n_{mi+\frac{1}{2},j} = \frac{v^n_{mi,j-\frac{1}{2}} + v^n_{mi,j+\frac{1}{2}} + v^n_{mi+1,j-\frac{1}{2}} + v^n_{mi+1,j+\frac{1}{2}}}{4}$$

$$K_1 = \frac{g\left(n_{i+\frac{1}{2},j}\right)^2\sqrt{\left(u^n_{mi+\frac{1}{2},j}\right)^2 + \left(v^n_{mi+\frac{1}{2},j}\right)^2}}{2\left(\frac{h^{n+1}_{i,j} + h^{n+1}_{i+1,j}}{2}\right)^{\frac{4}{3}}}$$

$$M^{n+2}_{i+\frac{1}{2},j} = \frac{\frac{M^n_{i+\frac{1}{2},j}}{2\Delta t} - \frac{\partial(u_m M)}{\partial x} - \frac{\partial(v_m M)}{\partial y} - gh\frac{\partial H}{\partial x} - K_1 M^n_{i+\frac{1}{2},j}}{\frac{1}{2\Delta t} + K_1}$$

$$u^n_{mi+\frac{1}{2},j+\frac{1}{2}} = \frac{4\left(M^n_{i+\frac{1}{2},j} + M^n_{i+\frac{1}{2},j+1}\right)}{h^{n-1}_{i,j}+h^{n-1}_{i,j+1}+h^{n-1}_{i+1,j}+h^{n-1}_{i+1,j+1}+h^{n+1}_{i,j}+h^{n+1}_{i,j+1}+h^{n-1}_{i+1,j}+h^{n-1}_{i+1,j+1}}$$

$$N^n_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{N^n_{i,j+\frac{1}{2}} + N^n_{i+1,j+\frac{1}{2}}}{2}$$

$$\frac{\partial(u_m N)}{\partial x} = \frac{u^n_{mi+\frac{1}{2},j+\frac{1}{2}}\overline{N^n_{i+\frac{1}{2},j+\frac{1}{2}}} - u^n_{mi-\frac{1}{2},j+\frac{1}{2}}\overline{N^n_{i-\frac{1}{2},j+\frac{1}{2}}}}{\Delta x}$$

$$v^n_{mi,j} = \frac{N^n_{i,j-\frac{1}{2}}+N^n_{i,j+\frac{1}{2}}}{h^{n-1}_{i,j}+h^{n+1}_{i,j}} \text{ dan } \overline{N^n_{i,j}} = \frac{N^n_{i,j-\frac{1}{2}}+N^n_{i,j+\frac{1}{2}}}{2}$$

$$\frac{\partial(v_m N)}{\partial y} = \frac{v^n_{mi,j+1}\overline{N^n_{i,j+1}} - v^n_{mi,j}\overline{N^n_{i,j}}}{\Delta y}$$

$$gh\frac{\partial H}{\partial y} = \frac{h^{n+1}_{i,j}+h^{n+1}_{i,j+1}}{2}\,\frac{Z^{n+1}_{bi,j+1}+h^{n+1}_{i,j+1}-Z^{n+1}_{bi,j}-h^{n+1}_{i,j}}{\Delta x}$$

$$u^n_{mi,j+\frac{1}{2}} = \frac{u^n_{mi-\frac{1}{2},j} + u^n_{mi-\frac{1}{2},j+1} + u^n_{mi+\frac{1}{2},j} + u^n_{mi+\frac{1}{2},j+1}}{4}$$

$$K_2 = \frac{g\left(n_{i,j+\frac{1}{2}}\right)^2\sqrt{\left(u^n_{mi,j+\frac{1}{2}}\right)^2+\left(v^n_{mi,j+\frac{1}{2}}\right)^2}}{2\left(\frac{h^{n+1}_{i,j}+h^{n+1}_{i,j+1}}{2}\right)^{\frac{4}{3}}}$$

$$N^{n+2}_{i,j+\frac{1}{2}} = \frac{N^n_{i,j+\frac{1}{2}} - \frac{\partial(u_m M)}{\partial x} - \frac{\partial(v_m M)}{\partial y} - gh\frac{\partial H}{\partial y} - K_2 N^n_{i,j+\frac{1}{2}}}{\frac{1}{2\Delta t}+K_2}$$

2.3 Perhitungan Debit Banjir

Perhitungan debit banjir rencana di Kali Putih dapat menggunakan metode Rasional. Perhitungan metode rasional menggunakan rumus sebagai berikut:

$$Q = \frac{1}{3.6} f.r.A$$

dengan

$$r = \left(\frac{R_{24}}{24}\right)\left(\frac{24}{t}\right)^{\frac{2}{3}}$$

$$t = \frac{L}{W}$$

$$W = 72\left(\frac{H}{L}\right)^{0.6}$$

Koefisien pengaliran (f) tergantung dari beberapa faktor antara lain jenis tanah, kemiringan, luas dan bentuk sungai. Besarnya nilai koefisien pengaliran dapat dilihat pada Tabel 2.1.

Tabel 2.1. Koefisien Pengaliran[4]

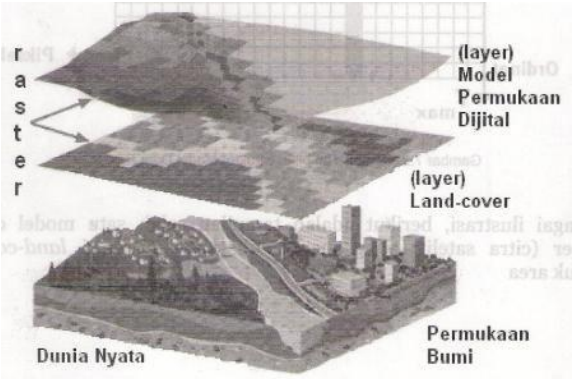
Kondisi Daerah Pengaliran	Koefisien Pengaliran (f)
Daerah pegunungan berlereng terjal	0,75-0,90
Daerah perbukitan	0,70-0,80
Tanah bergelombang dan semak-semak	0,50-0,75
Tanah daratan yang ditanami	0,45-0,65
Persawahan irigasi	0,70-0,80
Sungai di daerah pegunungan	0,75-0,85
Sungai kecil di daratan	0,45-0,75
Sungai besar yang setengah dari daerah pengalirannya terdiri dari daratan	0,50-0,75

Sumber: Sosrodarsono, 1989.

2.4 Data DEM

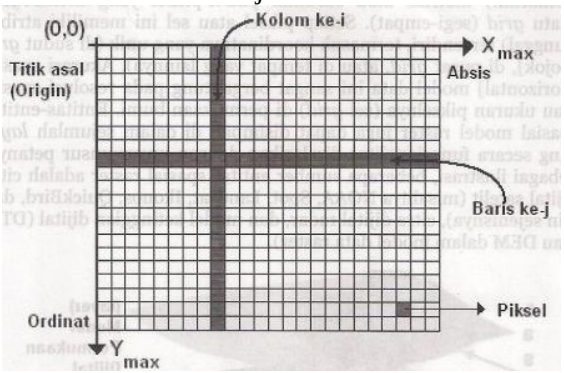
Data ketinggian yang akan digunakan dalam penelitian ini adalah data raster DEM-SRTM. DEM adalah data digital yang menggambarkan geometri dari bentuk permukaan bumi atau bagiannya yang terdiri dari himpunan titik-titik koordianat hasil sampling dari permukaan dengan algoritma yang didefinisikan permukaan tersebut menggunakan himpunan koordinat. DEM telah digunakan sebagai salah satu data utama untuk mendukung kegiatan pembuatan peta topografi, koreksi citra satelit, pemetaan daerah rawan bencana (banjir, tsunami, longsor, dan gunung api) dan penyusunan tataruang wilayah sehingga DEM sesuai untuk digunakan sebagai data pada simulasi aliran debris.

Kelebihan DEM SRTM adalah tingkat akurasi yang tinggi namun memiliki resolusi spasial yang rendah jika dibandingkan dengan DEM lain seperti DEM ALOS PRIS dan DEM dari peta topografi. Data DEM model layer raster adalah data yang memberikan informasi spasial permukaan bumi dengan merepresentasikannya sebagai elemen matriks atau sel-sel grid yang disebut dengan piksel. Unsur-unsur geografis (ketinggian) ditandai oleh nilai-nilai elemen matriks. Tampilan permukaan bumi dan DEM layer model raster ditunjukkan pada Gambar 2.4.



Gambar 2.4 Permukaan Bumi dan Layer Model Raster

Pada model data raster, matriks atau array dapat diurutkan menurut koordinat lokalnya yaitu kolom (x) dan baris (y). Selain itu, pada sistem koordinat piksel monitor komputer, secara default, titik asal sistem koordinat raster diletakkan di sudut kiri atas. Oleh karena itu, nilai absis (x) akan meningkat kearah kanan dan nilai ordinat (y) akan meningkat kearah bawah. Tampilan struktur model data raster ditunjukkan oleh Gambar 2.5.



Gambar 2.5 Struktur Model Data Raster

Selain data raster, juga terdapat data vektor. Data raster memiliki kelebihan dibandingkan dengan data vektor yaitu:

1. Memiliki struktur data yang sederhana
2. Secara teoritis, mudah dimanipulasi dengan menggunakan fungsi dan operator sederhana.
3. Compatible dengan citra-citra satelit penginderaan jauh dan semua image hasil scanning data spasial.
4. Memiliki kemampuan pemodelan dan analisis spasial tingkat lanjut.
5. Gambaran permukaan bumi dalam bentuk citra raster yang didapat dari sensor radar atau satelit, selalu lebih aktual dari pada bentuk vektornya.
6. Prosedur untuk memperoleh data dalam bentuk raster lebih mudah, sederhana dan murah[2].

2.5 Teknik Dekomposisi

Teknik dekomposisi merupakan metode yang digunakan untuk membagi proses komputasi dalam sebuah software menjadi bagian-bagian yang lebih kecil, dengan tujuan untuk mempermudah penyelesaiannya. Dekomposisi dapat digambarkan sebagai graf berarah dengan nodes merepresentasikan data dan edges merepresentasikan hubungan antar data. Terdapat empat teknik dekomposisi yaitu dekomposisi rekursif, dekomposisi data, dekomposisi exploratory, dan dekomposisi spekulatif. Dalam penelitian ini digunakan teknik dekomposisi data.

Dekomposisi data adalah metode yang baik dan umum digunakan untuk mendapatkan konkurensi dalam algoritma yang beroperasi pada struktur data besar. Dalam metode ini, dekomposisi dari proses komputasi dilakukan dalam dua langkah. Pada langkah pertama, data untuk proses komputasi dipartisi, dan pada langkah kedua, partisi data ini digunakan untuk menginduksi partisi dari proses komputasi. Dekomposisi data dibagi kedalam tiga jenis berdasarkan cara melakukan pertisi data, yaitu :

1. Partisi data *output*

   Dalam berbagai proses komputasi, setiap elemen *output* dapat diproses secara independen sebagai fungsi *input*. Dalam beberapa proses komputasi, partisi dari data *output* secara otomatis menginduksi dekomposisi dari permasalahan menjadi proses-proses(*tasks*), dimana setiap *task* diberikan tugas untuk mengerjakan proses komputasi sebagian dari *output*.

   Misalnya perkalian dua matriks A dan B berukuran n x n menjadi matriks C. Output pada matriks C dapat dipartisi kedalam empat sub-matriks seperti yang ditunjukkan pada Gambar 2.6. Sedangkan kemungkinan dekomposisi data *output* dari perkalian dua matriks tersebut digambarkan pada Tabel 2.2.

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} . \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

(a)

$$Task\ 1 : C_{1,1} = A_{1,1}B_{1,1} + A_{1,2} + B_{2,1}$$
$$Task\ 2 : C_{1,2} = A_{1,1}B_{1,2} + A_{1,2} + B_{2,2}$$
$$Task\ 3 : C_{2,1} = A_{2,1}B_{1,1} + A_{2,2} + B_{2,1}$$
$$Task\ 4 : C_{2,2} = A_{2,1}B_{1,2} + A_{2,2} + B_{2,2}$$

(b)

Gambar 2.6 (a) Partisi Matriks *Input* dan *Output* menjadi Sub Matriks 2 x 2 (b)Sebuah Dekomposisi Dari Perkalian Matriks Menjadi Empat *Task* Berdasarkan Pembagian Matriks(a)

Tabel 2.2 Dua Contoh Dekomposisi dari Perkalian Matriks menjadi Delapan *Task*

Dekomposisi I	Dekomposisi II
$Task\ 1: C_{1,1} = A_{1,1}.B_{1,1}$	$Task\ 1: C_{1,1} = A_{1,1}.B_{1,1}$
$Task\ 2: C_{1,1} = C_{1,1} + A_{1,2}.B_{2,1}$	$Task\ 2: C_{1,1} = C_{1,1} + A_{1,2}.B_{2,1}$
$Task\ 3: C_{1,2} = A_{1,1}.B_{1,2}$	$Task\ 3: C_{1,2} = A_{1,2}.B_{2,2}$
$Task\ 4: C_{1,2} = C_{1,2} + A_{1,2}.B_{2,2}$	$Task\ 4: C_{1,2} = C_{1,1} + A_{1,1}.B_{1,2}$
$Task\ 5: C_{2,1} = A_{2,1}.B_{1,1}$	$Task\ 5: C_{2,1} = A_{2,2}.B_{2,1}$
$Task\ 6: C_{2,1} = C_{2,1} + A_{2,2}.B_{2,1}$	$Task\ 6: C_{2,1} = C_{1,1} + A_{2,1}.B_{1,1}$
$Task\ 7: C_{2,2} = A_{2,1}.B_{1,2}$	$Task\ 7: C_{2,2} = A_{2,1}.B_{1,2}$
$Task\ 8: C_{2,2} = C_{2,2} + A_{2,2}.B_{2,2}$	$Task\ 8: C_{2,2} = C_{2,2} + A_{2,2}.B_{2,2}$

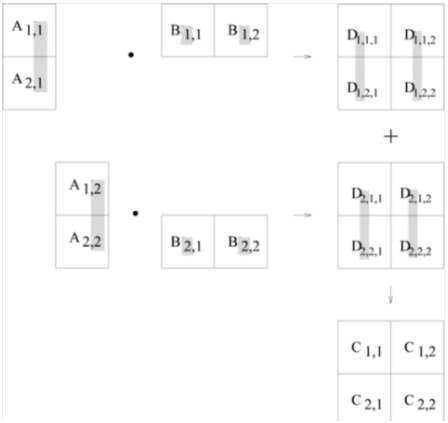
2. Partisi data *input*
   Partisi data *output* dapat dijalankan apabila setiap *output* dapat dihitung sebagai fungsi *input*. Dalam bebrapa algoritma, tidak mungkin untuk menjalankan partisi data *output*. Misalnya menentukan nilai minimum dan maksimum data. Dalam beberapa hal, dimungkinkan untuk melakukan partisi data *input*, kemudian menggunakan partisi ini untuk menginduksi konkurensi. *Task* dibentuk untuk setiap partisi dari data *input*

dan *task* ini mengerjakan proses komputasi sebanyak mungkin dengan menggunakan data lokal. Hasil dari *task* yang diperoleh dari partisi data *input* tidak langsung mrnyelesaikan permasalahan sebenarnya, proses komputasi selanjutnya dibutuhkan untuk mengkombinasikan hasilnya. Misalnya, ketika mencari jumlahan dari barisan N bilangan dengan menggunakan p proses ($N > p$), maka dapat dilakukan partisi *input* kedalam p bagian dengan ukuran yang hampir sama. Setiap *task* kemudian menghitung jumlah dari bilangan dalam sebuah bagian. Pada akhirnya, sebanyak p hasil partial dapat dijumlahkan untuk mendapatkan hasil akhirnya.

3. Partisi data *intermediate*

   Algoritma sering disusun sebagai proses komputasi multi-tahap sehingga *output* dari satu tahap merupakan masukan ke tahap berikutnya. Dekomposisi dari algoritma tersebut dapat diturunkan dengan membagi *input* atau *output* data dari tahap peralihan (*intermediate stage*) dari algoritma. Seringkali data *intermediate* tidak dihasilkan secara eksplisit dalam rangkaian algoritma untuk menyelesaikan masalah dan beberapa perubahan struktur algoritma asli mungkin dibutuhkan untuk menjalankan partisi data *intermediate*.

   Dekomposisi yang diinduksi oleh partisi 2 x 2 dari *output* pada matriks C pada Gambar 2.6 dan Tabel 2.2 memiliki derajat maksimum konkurensi empat. derajat konkurensi dapat dinaikkan dengan memberikan tahap *intermediate* dimana delapan *task* melakukan proses komputasi dari perkalian sub-matriksnya dan menyimpan hasilnya dalam sebuah matriks tiga dimensi sementara D, seperti yang ditunjukkan pada Gambar 2.7. Sub matriks $D_{k,i,j}$ adalah hasil perkalian dari $A_{i,k}$ dan $B_{k,j}$.



Gambar 2.7 Perkalian Matriks A dan B dengan Dekomposisi Matriks *Intermediate* Tiga Dimensi D.

Sebuah partisi dari matriks *intermediate* D digambarkan pada Gambar 2.8, sedangkan dekomposisi data *intermediate* kedalam duabelas *task* ditunjukkan pada Gambar 2.9. Setelah fase perkalian, tahap penjumlahan matriks yang relative murah dapat menghasilkan matriks hasil C. Semua submatriks $D_{*,i,j}$ dengan dimensi kedua dan ketiga yang sama i dan j ditambahkan untuk menghasilkan $C_{i,j}$. Graf dependensi *task* dari dekomposisi ditunjukkan oleh Gambar 2.10.

Stage I

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} \begin{pmatrix} D_{1,1,1} & D_{1,1,2} \\ D_{1,2,2} & D_{1,2,2} \end{pmatrix} \\ \begin{pmatrix} D_{2,1,1} & D_{2,1,2} \\ D_{2,2,2} & D_{2,2,2} \end{pmatrix} \end{pmatrix}$$

Stage II

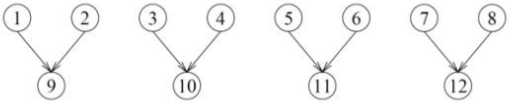
$$\begin{pmatrix} D_{1,1,1} & D_{1,1,2} \\ D_{1,2,2} & D_{1,2,2} \end{pmatrix} + \begin{pmatrix} D_{2,1,1} & D_{2,1,2} \\ D_{2,2,2} & D_{2,2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Gambar 2.8 (Tahap I) Partisi Matriks *Input* menjadi Matriks *Intermediate* Tiga Dimensi D, (Tahap II) Penjumlahan Matriks *Intermediate* untuk Mendapatkan Matriks Hasil C.

A decomposition induced by a partitioning of D

$$
\begin{aligned}
\text{Task 01:} \quad & D_{1,1,1} = A_{1,1}B_{1,1} \\
\text{Task 02:} \quad & D_{2,1,1} = A_{1,2}B_{2,1} \\
\text{Task 03:} \quad & D_{1,1,2} = A_{1,1}B_{1,2} \\
\text{Task 04:} \quad & D_{2,1,2} = A_{1,2}B_{2,2} \\
\text{Task 05:} \quad & D_{1,2,1} = A_{2,1}B_{1,1} \\
\text{Task 06:} \quad & D_{2,2,1} = A_{2,2}B_{2,1} \\
\text{Task 07:} \quad & D_{1,2,2} = A_{2,1}B_{1,2} \\
\text{Task 08:} \quad & D_{2,2,2} = A_{2,2}B_{2,2} \\
\text{Task 09:} \quad & C_{1,1} = D_{1,1,1} + D_{2,1,1} \\
\text{Task 10:} \quad & C_{1,2} = D_{1,1,2} + D_{2,1,2} \\
\text{Task 11:} \quad & C_{2,1} = D_{1,2,1} + D_{2,2,1} \\
\text{Task 12:} \quad & C_{2,2} = D_{1,2,2} + D_{2,2,2}
\end{aligned}
$$

Gambar 2.9 Dekomposisi dari Perkalian Matriks Berdasarkan Partisi dari Matriks Tiga Dimensi *Intermediate*



Gambar 2.10 Graf Dependensi *Task* dari Dekomposisi

“Halaman ini sengaja dikosongkan”

BAB III
METODOLOGI PENELITIAN

Bab ini menjelaskan metodologi yang diterapkan untuk menyelesaikan permasalahan dalam Tugas Akhir. Metodologi penelitian berisikan objek penelitian dan tahap penelitian.

3.1 Objek Penelitian

Objek penelitian dalam Tugas Akhir ini adalah aliran debris di Kali Putih, Gunung Merapi dengan spesifikasi sebagai berikut :

1. Lahar yang dikeluarkan dari Gunung Merapi pada letusan tahun 2010 adalah 12.000.000 m^3.
2. Kedalaman aliran lahar (*h*) adalah 12 m.
3. Kecepatan aliran lahar saat mengalir adalah 65 km/jam.

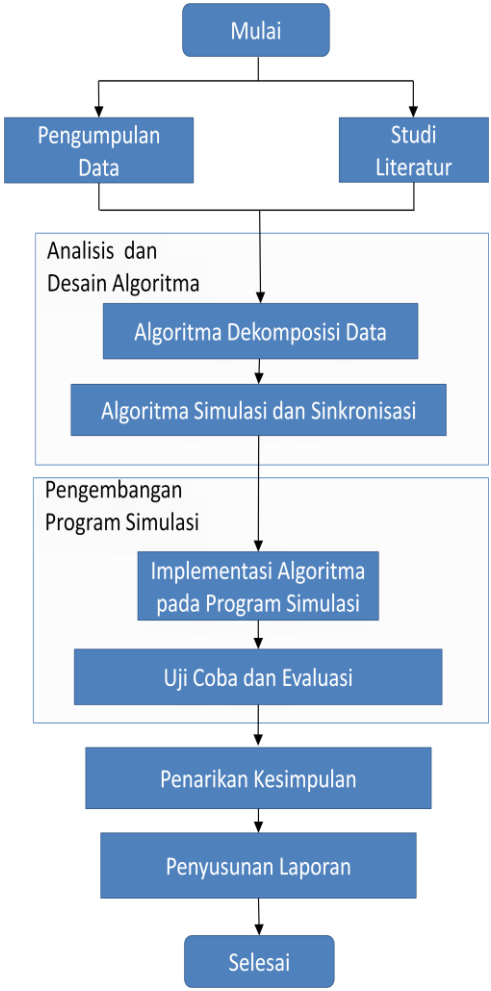
3.2 Tahap Penelitian

Tahap-tahap yang dilakukan dalam penyusunan Tugas Akhir ini adalah sebagai berikut:

1. Studi Literatur
   Pada tahap pertama ini meliputi identifikasi permasalahan dan mencari refrensi mengenai pengolahan data *Digital Elevation Model* (DEM), teknik dekomposisi data, dan simulasi penyebaran aliran debris satu dimensi dan dua dimensi.
2. Pengumpulan data
   Pada tahap ini dilakukan pengumpulan data yang terkait pada permasalahan Tugas Akhir. Data yang akan digunakan dalam penelitian ini adalah data DEM yang diperoleh dari Tugas Akhir Belgis Ainatul Iza yang berjudul "Penyelesaian Numerik Dari Model Penyebaran Aliran Debris Satu Dan Dua Dimensi Dengan Metode Beda Hingga".

3. Analisis dan Perancangan Sistem
   Pada tahap ini akan dilakukan analisis bentuk data DEM yang diperoleh dan analisis teknik dekomposisi data yang digunakan. Kemudian dilanjutkan dengan perancangan tahap-tahap dalam dekomposisi data DEM dan simulasi aliran debris.
4. Implementasi pada perangkat lunak simulasi
   Pada tahap ini, akan dilakukan implementasi pada perangkat lunak simulasi aliran debris berdasarkan hasil analisis dan perancangan sistem. Data DEM yang telah diperoleh dibagi menjadi data dengan ukuran lebih kecil (sub-matriks). Selanjutnya sub-matriks tersebut dengan variabel-variabel lain digunakan dalam proses simulasi. Simulasi pertama dijalankan pada sub matriks pertama bersama variabel lain hingga diperoleh hasil simulasi berupa gambaran aliran debris dan dampaknya pada lereng pertama serta informasi tambahan berupa sub-matriks selanjutnya yang akan diakses. Simulasi dilanjutkan dengan menggunakan data sub-matriks selanjutnya. Implementasi perangkat lunak simulasi ini menggunakan MATLAB.
5. Pengujian dan Evaluasi
   Pada tahap ini dilakukan pengujian dan evaluasi terhadap perangkat lunak yang telah dibuat. Pengujian meliputi proses perangkat lunak tersebut bekerja serta hasil yang ditampilkan.
6. Penarikan Kesimpulan
   Pada tahap ini dilakukan penarikan kesimpulan terhadap hasil dekomposisi data DEM dalam simulasi model penyebaran aliran debris satu dan dua dimensi dan saran yang diperlukan untuk penelitian selanjutnya.

Tahap-tahap pengerjaan Tugas Akhir yang telah dijelaskan di atas digambarkan dalam diagram alir pada Gambar 3.1.



Gambar 3.1. Diagram alir metode penelitian

“Halaman ini *sengaja di kosongkan”*

BAB IV
PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai perancangan dan hasil implementasi perangkat lunak dalam Tugas Akhir ini. Perancangan berisikan perancangan data dan perancangan algoritma. Implementasi meliputi Implementasi antarmuka, implementasi pengolahan DEM dan implementasi simulasi aliran.

4.1 Analisis

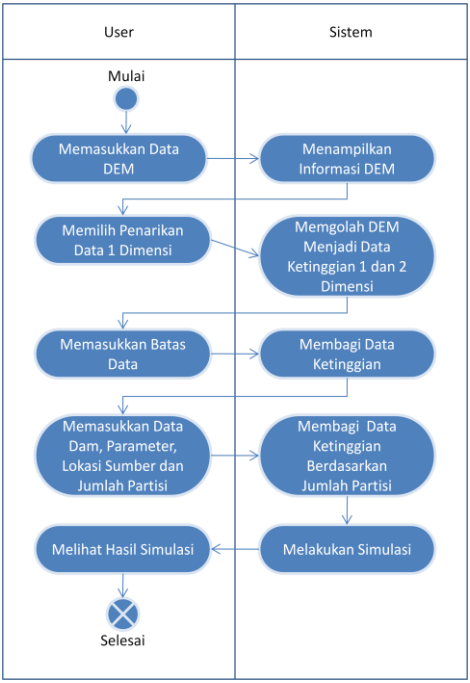
Sebelum melaksanakan perancangan, dibutuhkan langkah awal dalam Tugas Akhir yaitu melakukan analisis perangkat lunak dan analisis kebutuhan. Analisis perangkat lunak menunjukkan tahapan utama dalam perangkat lunak, sedangkan analisis kebutuhan menunjukkan perngkat keras dan perangkat lunak yang dibutuhkan dalam perancanagan dan implementasi.

4.1.1 Analisis Perangkat Lunak

Perangkat lunak yang dibangun diharapkan dapat melakukan simulasi model penyebaran aliran debris satu dan dua dimensi dengan menggunakan data DEM dan memiliki waktu komputasi yang lebih rendah. *Swimlane Diagram* yang menunjukkan tahapan-tahapan dari perangkat lunak disajikan pada Gambar 4.1. Penjelasan *swimlane diagram* perangkat lunak adalah sebagai berikut:

1. *User* memasukkan data DEM kedalam perangkat lunak.
2. Sistem akan menampilkan informasi mengenai data DEM yang telah dimasukkan. Informasi tersebut meliputi koordinat data, ketinggian minimum data, ketinggian maksimum data dan jarak antar data ketinggian.
3. *User* memasukkan cara penarikan data ketinggian satu dimensi. *User* dapat memilih cara penarikan data berdasarkan nilai rata-rata, minimum, atau maksimum pada sumbu-x atau sumbu-y.

4. Sistem mengolah data DEM menjadi data ketinggian satu dimensi dan dua dimensi berdasarkan cara penarikan data yang telah dipilih *user*. Setelah diperoleh data ketinggian satu dan dua dimensi, sistem akan menampilkan kontur data tersebut.
5. *User* memasukkan batas data satu dan dua dimensi.
6. Sistem akan membagi data ketinggian pada batas yang telah ditentukan *user*.
7. *User* memasukkan data dam, parameter aliran debris, lokasi sumber, dan jumlah partisi.
8. Sistem membagi data ketinggian berdasarkan jumlah partisi kemudian menjalankan dekomposisi data dalam simulasi aliran debris.

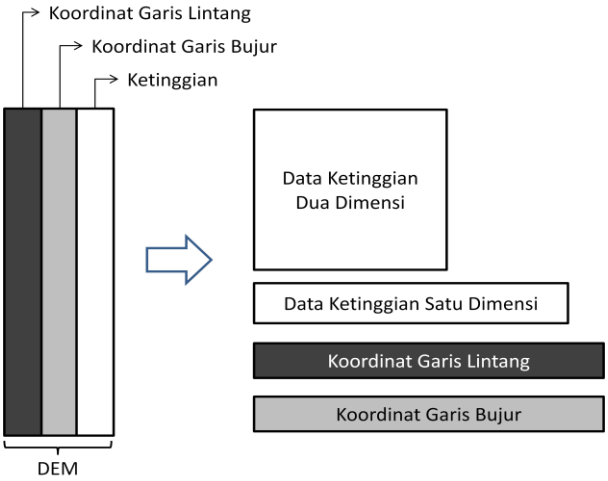


Gambar 4.1 *Swimlane Diagram* Simulasi Aliran Debris Satu dan Dua Dimensi

Tahapan dalam perangkat lunak ini adalah sebagai berikut :

1. Pengolahan Data DEM

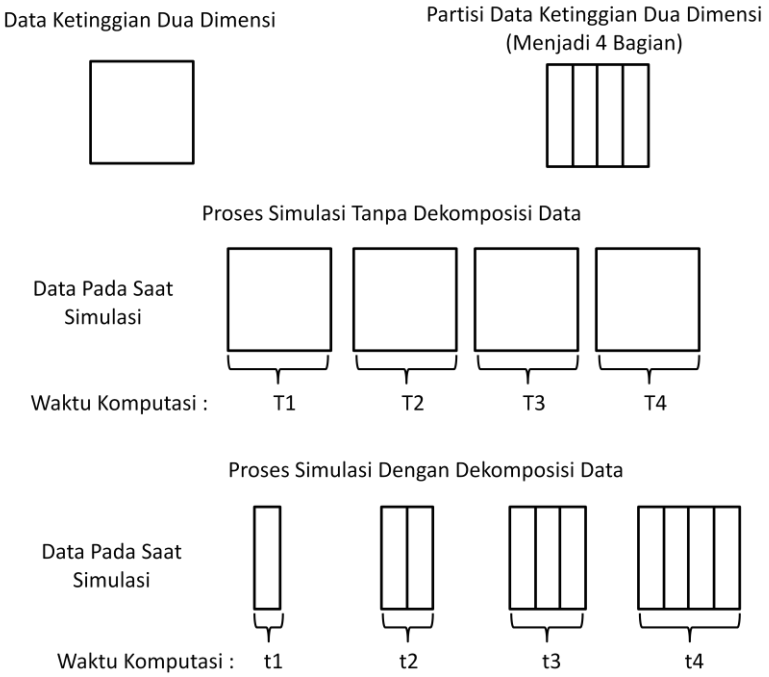
Bentuk data DEM dalam penelitian ini ketika dibuka dengan menggunakan MATLAB adalah matriks tiga kolom, dengan kolom pertama dan kedua masing-masing merupakan kolom yang menunjukkan koordinat garis lintang dan garis bujur pada peta, sedangkan kolom ketiga menunjukkan ketinggian tempat dari koordinat tersebut. Data dalam bentuk ini lebih sulit ditampilkan dan diolah untuk digunakan pada simulasi aliran debris sehingga data DEM sehingga perlu untuk diubah kedalam bentuk data yang lebih sederhana yaitu dalam bentuk matriks satu dimensi dan dua dimensi. Data ketinggian dari DEM (nilai pada kolom ketiga) disimpan dalam bentuk matriks satu dan dua dimensi sedangkan koordinat data (nilai pada kolom pertama dan kedua) disimpan dalam matriks satu dimensi lainnya. Bentuk data DEM dan bentuk hasil pengolahan data ditunjukkan pada Gambar 4.2 berikut ini.



Gambar 4.2 Bentuk Data DEM dan Hasil Pengolahan Data DEM

2. Dekomposisi Data

Proses dekomposisi data dijalankan pada data DEM yang telah diolah menjadi matriks data ketinggian dua dimensi. Seperti teknik dekomposisi data pada umumnya, pada langkah pertama, data ketinggian dipartisi dan pada langkah kedua, setelah partisi data pertama selesai digunakan kemudian diperoleh partisi data selanjutnya, partisi data sebelumnya dan partisi data selanjutnya digabungkan untuk proses simulasi berikutnya. Proses dekomposisi data dalam program ini diilustrasikan pada Gambar 4.3 berikut ini.



Gambar 4.3 Proses Dekomposisi Data

Karena besarnya data yang diolah berbanding lurus dengan lama proses pengolahan terhadap data tersebut (besarnya waktu komputasi), maka dapat diperoleh

$$t1 < T1, t2 < T2, t3 < T3 \text{ dan } t4 = T4$$

sehingga

$$WD = t1 + t2 + t3 + t4 < T1 + T2 + T3 + T4 = WTD$$
$$WD < WTD$$

dengan,
$T1, T2, T3, T4$ = Waktu Komputasi Tanpa Dekomposisi Data Pertama, Kedua , Ketiga, Keempat.
$t1, t2, t3, t4$ = Waktu Komputasi Dengan Dekomposisi Data Pertama, Kedua , Ketiga, Keempat.
WTD = Total Waktu Komputasi Tanpa Dekomposisi Data
WD = Total Waktu Komputasi Dengan Dekomposisi Data

Dengan demikian dekomposisi data dengan cara ini diharapkan dapat menurunkan waktu komputasi yang dibutuhkan dalam melakukan simulasi.

4.1.2 Analisis Kebutuhan

Kebutuhan dalam perancangan dan implementasi simulasi aliran debris satu dan dua dimensi meliputi perangkat keras dan lunak komputer. Detail dari perangkat keras dan lunak yang digunakan dapat dilihat pada Tabel 4.1 berikut.

Tabel 4.1 Kebutuhan Perancangan dan Implementasi Sistem

Perangkat Keras	Prosesor : Intel(R) Core(TM) i3 – 2350M CPU @2.3GHz
	Memory : 2 GB
Perangkat Lunak	Sistem Operasi : Windows 7 Ultimate 32-bit
	Tools : MATLAB 7.8.0 (R2009a)

4.2 Perancangan

Setelah analisis dilaksanakan maka kemudian dilanjutkan dengan perancangan. Perancangan meliputi perancangan data dan perancangan proses algoritma.

4.2.1 Perancangan Data

Terdapat tiga macam data dalam Tugas Akhir ini yaitu data masukan, data proses dan data keluaran. Data masukan adalah data DEM. Data proses merupakan data parameter-parameter yang digunakan dalam preoses simulasi. Data keluaran adalah gambaran simulasi aliran debris satu dan dua dimensi dan waktu komputasi.

1. Data Masukan

Data masukan dalam penelitian ini adalah data DEM SRTM yang diperoleh dari BPPTK Yogyakarta. Data yang diperoleh ini memiliki format .xyz.

2. Data Proses

Data proses adalah data yang digunakan dalam proses pengolahan data masukan. Data proses tersebut adalah sebagai berikut :

a. Data dam yakni jarak antara titik pusat dam dan puncak gunung (Ld), jarak antara titik pusat dam dan titik awal dam ($B1$), jarak antara titik pusat dam dan titik akhir dam ($B2$) dan ketinggian dam (Hd).
b. Parameter-parameter yaitu panjang sungai(L), jumlah pias, diameter sedimen (d), durasi waktu (T), konsentrasi sedimen (Cd), debit (Q), kedalaman aliran saat di puncak (h), tinggi lereng (Z_b) dan lama simulasi.
c. Lokasi sumber yang meliputi arah dan jarak.
d. Jumlah partisi data.

3. Data Keluaran

Data Keluaran pada simulasi aliran debris ini adalah gambaran aliran debris yang menuruni Kali Putih Gunung Merapi beserta waktu komputasi yang menunjukkan kecepatan komputasi simulasi ketika menggunakan dekomposisi data.

4.2.2 Perancangan Algoritma

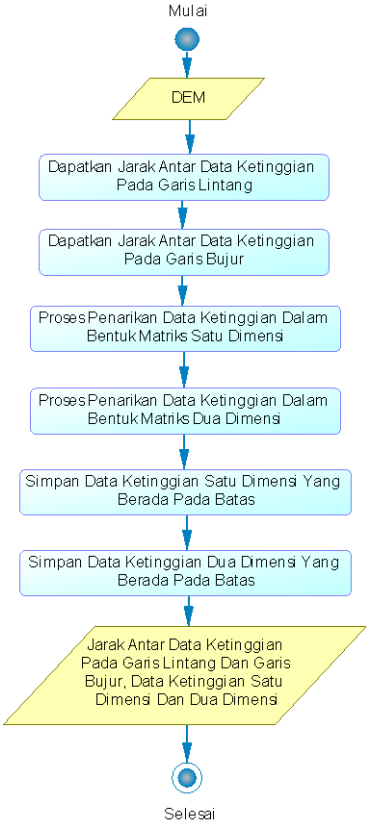
Perancangan algoritma ini diawali dengan pengolahan DEM menjadi data ketinggian satu dan dua dimensi. Proses berikutnya adalah pengolahan data simulasi berupa data ketinggian, data dam, parameter, lokasi sumber, dan jumlah partisi. Data yang telah diolah dari tahap sebelumnya kemudian digunakan pada proses simulasi aliran debris satu dan dua dimensi dengan menerapkan proses dekomposisi data didalamnya.

1. Proses Pengolahan Data DEM

Tahap pertama adalah perancangan proses pengolahan data *Digital Elevation Model* (DEM). Dalam proses ini data DEM diubah menjadi data ketinggian satu dan dua dimensi. Data ketinggian satu dimensi diperoleh dengan menghitung nilai rata-rata, minimum atau maksimum dari data DEM berdasarkan sumbu-x atau sumbu-y. Data ketinggian dalam bentuk matriks dua dimensi diperoleh dengan meletakkan data-data ketinggian dari data DEM dalam sebuah matriks dua dimensi. Peletakan data ketinggian didasarkan pada koordinat-koordinat data tersebut.

Selain data ketinggian satu dan dua dimensi, juga dicari jarak antar data ketinggian berdasarkan koordinat garis lintang dan garis bujur. Setelah mengetahui jarak antar data ketinggian maka koordinat-koordinat dari data DEM tidak disimpan semuanya, namun cukup disimpan koordinat terkecil pada sumbu-x dan sumbu-y. Koordinat terkecil dan jarak antar data ketinggian ini digunakan sebagai acuan untuk menentukan nilai koordinat-koordinat dari semua data ketinggian satu dan dua dimensi.

Berdasarkan interpretasi fisik simulasi aliran debris satu dan dua dimensi maka data yang diperoleh harus dipotong dimana data satu dimensi berada pada lereng bagian atas dan data dua dimensi pada lereng bagian bawah. Oleh karena itu, pengguna harus memberikan batas lokasi antara data satu dimensi dan data dua dimensi. Setelah data ketinggian satu dan dua dimensi di potong berdasarkan batas yang telah ditentukan maka data ketinggian satu dan dua dimensi, jarak antar data ketinggian dan koordinat terkecil disimpan. Gambar 4.4 berikut menunjukkan *activity diagram* proses pengolahan DEM.

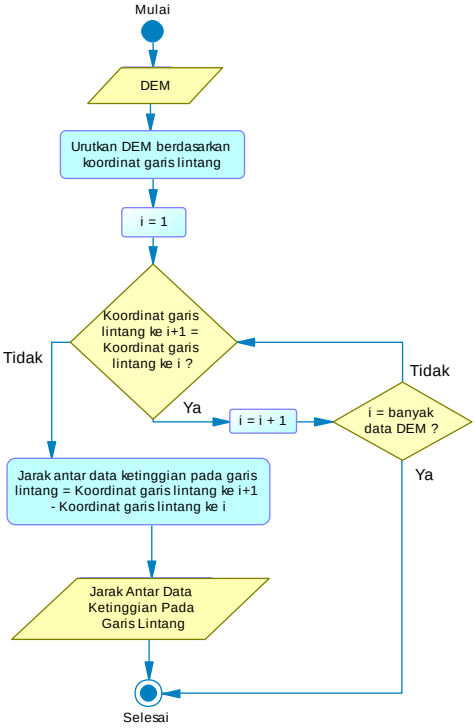


Gambar 4.4 *Activity Diagram* Pengolahan Data DEM

Algoritma pencarian jarak antar data pada sumbu-x adalah sebagai berikut :

1. Urutkan data DEM berdasarkan kolom pertama yang merupakan koordinat garis lintang data DEM.
2. Pada kolom pertama, bandingkan data pertama dengan data kedua, apabila data sama maka bandingkan data kedua dan data ketiga dan seterusnya hingga diperoleh dua buah data yang berbeda. Selisih kedua data ini adalah jarak antar data pada sumbu-x.

Diagram alir algoritma pencarian jarak antar data pada sumbu-x ditunjukkan oleh Gambar 4.5 berikut ini.

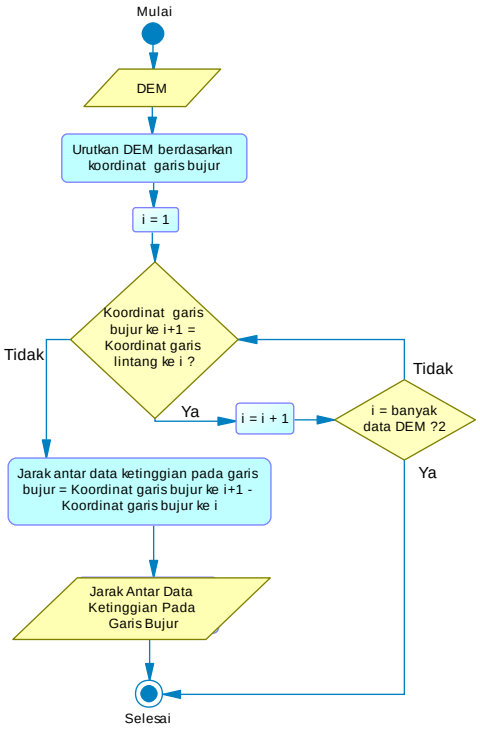


Gambar 4.5 Algoritma Pencarian Jarak Antar Data Ketinggian pada Sumbu-x

Algoritma pencarian jarak antar data pada sumbu-y adalah sebagai berikut :

1. Urutkan data DEM berdasarkan kolom kedua yang merupakan koordinat garis bujur data DEM.
2. Pada kolom kedua, bandingkan data pertama dengan data kedua, apabila data sama maka bandingkan data kedua dan data ketiga dan seterusnya hingga diperoleh dua data yang berbeda. Selisih kedua data ini adalah jarak antar data pada sumbu-y.

Algoritma pencarian jarak antar data pada sumbu-y ditunjukkan oleh Gambar 4.6 berikut ini.

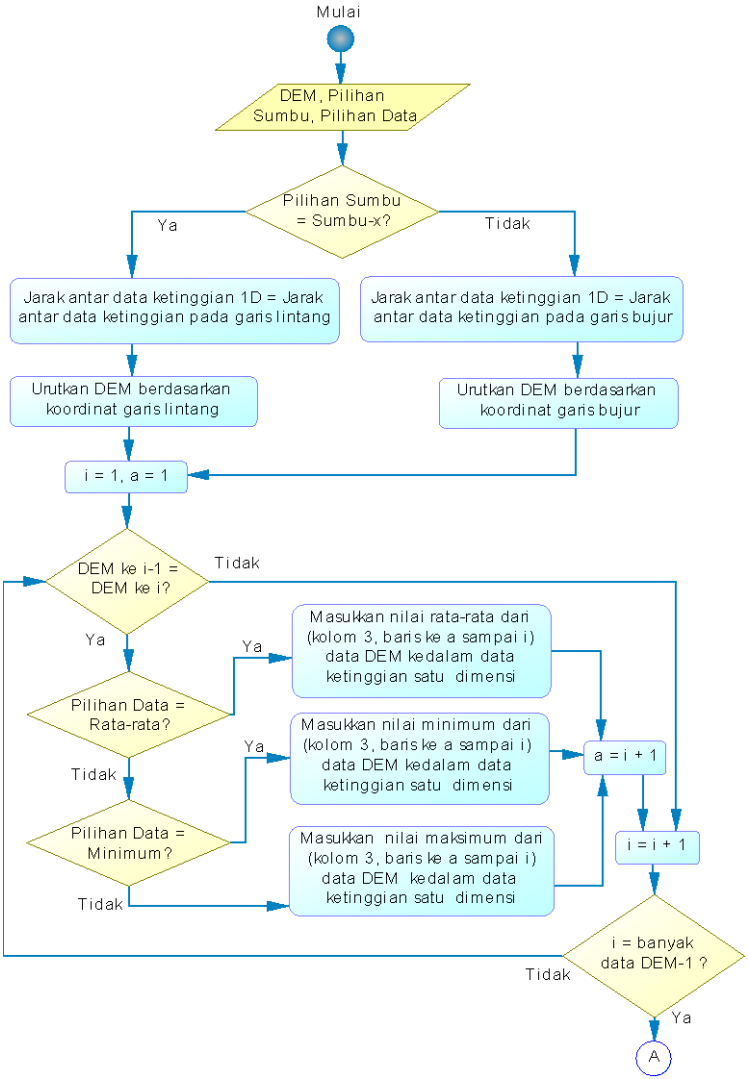


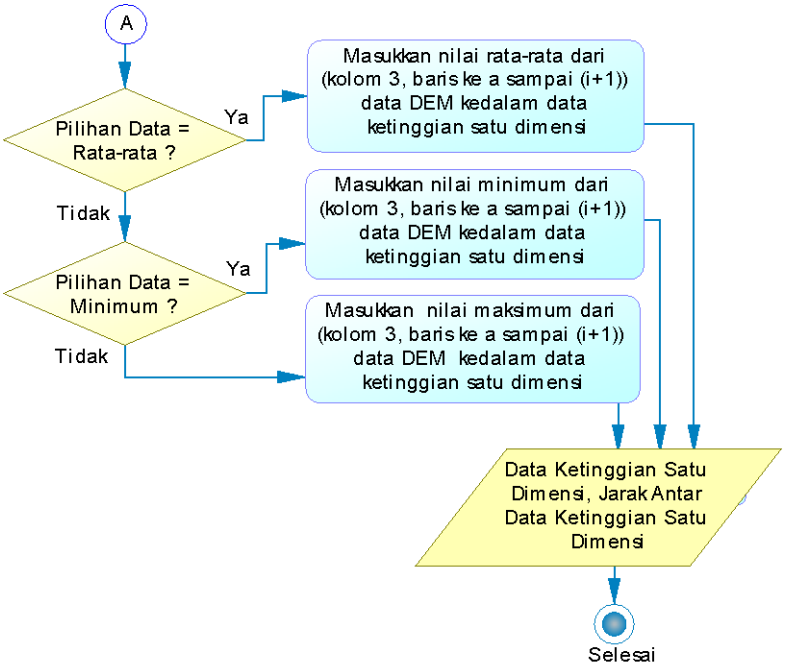
Gambar 4.6 Algoritma Pencarian Jarak Antar Data Ketinggian pada Sumbu-y

Algoritma penarikan data ketinggian dalam bentuk matriks satu dimensi adalah sebagai berikut :

1. Tentukan jarak antar data ketinggian satu dimensi. Jarak antar data ketinggian satu dimensi ditentukan berdasarkan pilihan sumbu yang dimasukkan oleh pengguna. Jarak antar data ketinggian satu dimensi adalah jarak antar data ketinggian pada garis lintang apabila pengguna memilih penarikan data satu dimensi berdasarkan sumbu-x. Sebaliknya, Jarak antar data ketinggian satu dimensi adalah jarak antar data ketinggian pada garis bujur apabila pengguna memilih penarikan data satu dimensi berdasarkan sumbu-y.
2. Urutkan DEM berdasarkan koordinat garis lintang apabila pengguna memilih penarikan data satu dimensi berdasarkan sumbu-x. Sebaliknya, urutkan DEM berdasarkan koordinat garis bujur apabila pengguna memilih penarikan data satu dimensi berdasarkan sumbu-y.
3. Jalankan perulangan *for* untuk proses pengisian data ketinggian satu dimensi. Proses pengisian data ketinggian satu dimensi ini berdasarkan pilihan penarikan data oleh pengguna. Apabila pengguna memilih penarikan data menggunakan nilai rata-rata maka nilai rata-rata dari koordinat garis lintang (apabila penarikan data satu dimensi berdasarkan sumbu-x) atau garis bujur (apabila penarikan data satu dimensi berdasarkan sumbu-y) yang sama dimasukkan kedalam matriks satu dimensi data ketinggian. Demikian juga, apabila pengguna memilih penarikan data menggunakan nilai minimum atau maksimum maka nilai minimum atau maksimum dari koordinat garis lintang atau garis bujur yang sama dimasukkan kedalam matriks satu dimensi data ketinggian.

Algoritma penarikan data ketinggian dalam bentuk matriks satu dimensi ditunjukkan oleh Gambar 4.7 berikut ini.



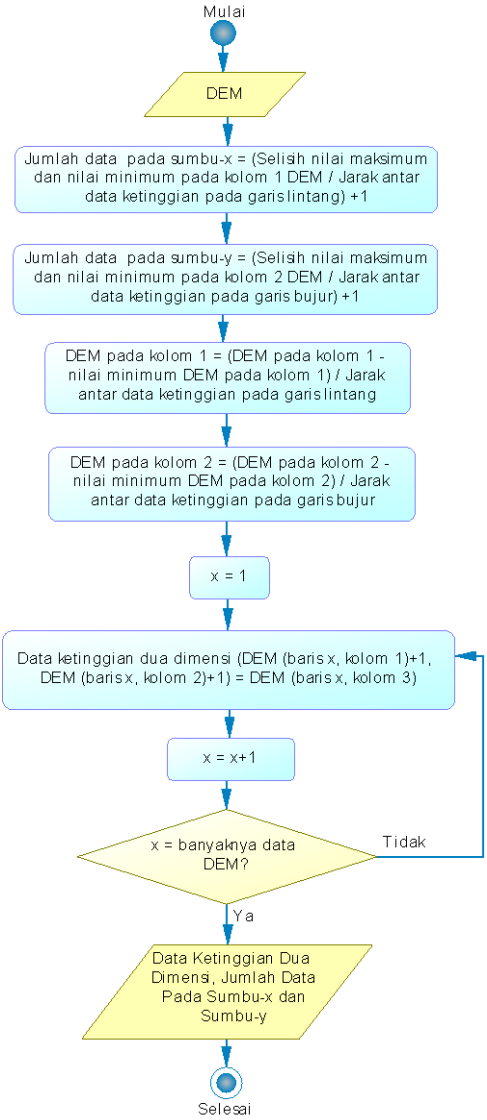


Gambar 4.7 Algoritma Penarikan Data Ketinggian Dalam Bentuk Matriks Satu Dimensi

Algoritma penarikan data ketinggian dalam bentuk matriks dua dimensi adalah sebagai berikut:

1. Dapatkan jumlah data pada sumbu-x. Jumlah data pada sumbu-x ini diperoleh dengan cara membagi selisih dari nilai maksimum dan minimum pada kolom pertama DEM (yang berisi koordinat garis lintang) dengan jarak antar data ketinggian pada garis lintang.
2. Dapatkan jumlah data pada sumbu-y. Jumlah data pada sumbu-y ini diperoleh dengan cara membagi selisih dari nilai maksimum dan minimum pada kolom kedua DEM (yang berisi koordinat garis bujur) dengan jarak antar data ketinggian pada garis bujur.
3. Ubah nilai DEM pada kolom pertama dengan cara menguranginya dengan nilai minimum pada kolom ini dan membaginya dengan jarak antar data ketinggian pada garis lintang.
4. Ubah nilai DEM pada kolom kedua dengan cara menguranginya dengan nilai minimum pada kolom ini dan membaginya dengan jarak antar data ketinggian pada garis bujur.
5. Jalankan perulangan *for* sebanyak jumlah data DEM (dengan nilIai x dimulai dari 1 sampai panjang data DEM). Data ketinggian dua dimensi baris a, kolom b diisi dengan nilai DEM pada baris ke-x kolom ketiga. Dengan a adalah nilai DEM pada baris ke-x kolom pertama ditambah satu dan b adalah nilai DEM pada baris ke-x kolom kedua ditambah satu.

Algoritma penarikan data ketinggian dalam bentuk matriks satu dimensi ditunjukkan oleh Gambar 4.8.

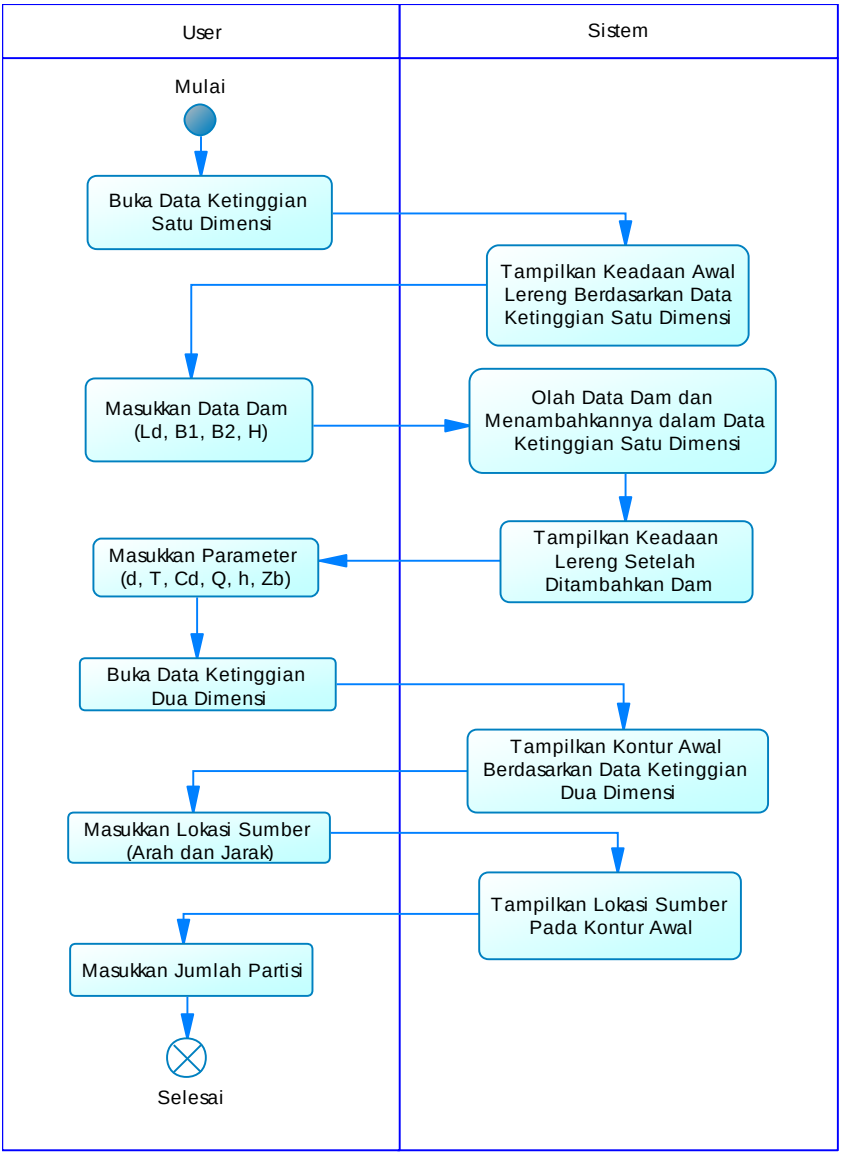


Gambar 4.8 Algoritma Penarikan Data Ketinggian Dalam Bentuk Matriks Dua Dimensi

2. Proses Pengolahan Data Simulasi

Proses pengolahan data simulasi bertujuan untuk mendapatkan data-data simulasi yang dibutuhkan dan mengolahnya sebelum digunakan dalam proses simulasi aliran debris. Data-data tersebut adalah data ketinggian satu dan dua dimensi, data dam, parameter-parameter simulasi, jumlah partisi, dan lokasi sumber.

Data ketinggian satu dan data ketinggian dua dimensi adalah data-data yang diperoleh dari proses sebelumnya. Data sabo dam terdiri atas jarak antara titik pusat dam dan puncak gunung (Ld), jarak antara titik pusat dam dan titik awal dam ($B1$), jarak antara titik pusat dam dan titik akhir dam ($B2$) dan ketinggian dam (Hd) yang dibutuhkan untuk simulasi aliran debris satu dimensi. Selain itu juga dimasukan parameter-parameter yang terdiri atas diameter sedimen (d), durasi waktu (T), konsentrasi sedimen (Cd), debit (Q), kedalaman aliran saat di puncak (h), dan tinggi lereng (Zb). Debit dapat dimasukkan secara langsung atau melalui perhitungan debit dengan cara memasukkan parameter luas sungai (A), panjang sungai (L), beda tinggi hulu hilir, curah hujan harian ($R24$) dan koefisien pengaliran yang didasarkan pada kondisi daerah pengaliran dan sungai (f). Data lokasi sumber dalam simulasi aliran debris dua dimensi terdiri atas arah aliran dan jarak (jarak yang dimaksud adalah jarak koordinat tersebut dari sumbu-x atau sumbu-y). Jumlah partisi adalah banyaknya partisi data dalam proses dekomposisi data simulasi aliran debris dua dimensi. Gambar 4.9 berikut menunjukkan *swimlane diagram* proses pengolahan data simulasi.

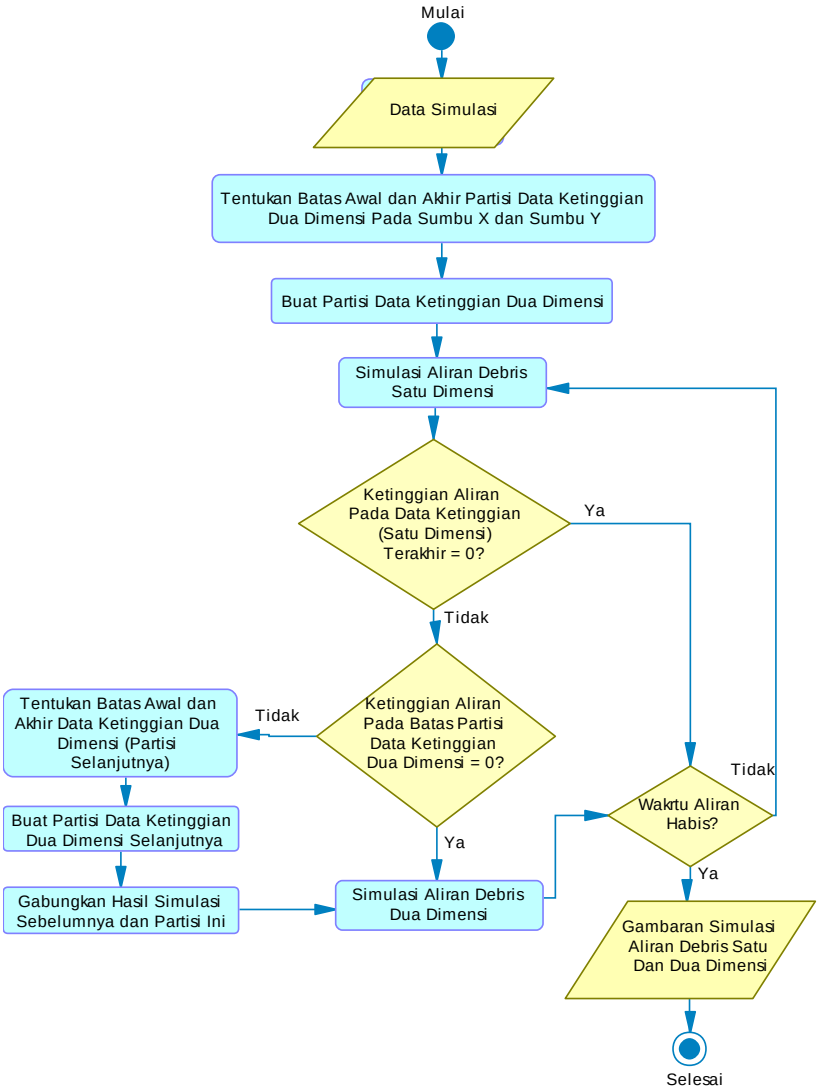


Gambar 4.9 *Swimlane Diagram* Pengolahan Data Simulasi

3. Proses Simulasi Aliran

Proses simulasi aliran adalah proses terakhir. Proses ini bertujuan untuk menggambarkan aliran debris di Kali Putih Gunung Merapi. Didalam simulasi aliran debris satu dimensi dan dua dimensi ini peneliti melakukan dekomposisi data untuk mendapatkan hasil simulasi dengan waktu komputasi yang lebih cepat.

Proses ini diawali dengan mendapatkan pertisi pertama data ketinggian dua dimensi. Selanjutnya, proses simulasi aliran debris satu dimensi dijalankan. Setelah semua data ketinggian satu dimensi telah digunakan dalam proses simulasi aliran debris satu dimensi maka proses simulasi aliran debris dua dimensi dijalankan dengan menggunakan partisi data ketinggian dua dimensi pertama. Saat proses simulasi aliran debris dua dimensi mencapai batas partisi data maka data partisi sebelumnya digabungkan dengan data partisi selanjutnya, selanjutnya proses simulasi aliran debris dua dimensi dilanjutkan dengan menggunakan data partisi gabungan ini. Proses simulasi aliran debris satu dimensi tetap berjalan ketika simulasi aliran debris dua dimensi berjalan. Ketika waktu simulasi habis, diperoleh hasil simulasi berupa aliran debris satu dimensi dan dua dimensi pada periode waktu yang telah ditentukan dan waktu komputasi yang dibutuhkan. Gambar 4.10 berikut menunjukkan diagram alir proses simulasi.



Gambar 4.10 *Activity Diagram* Proses Simulasi Aliran

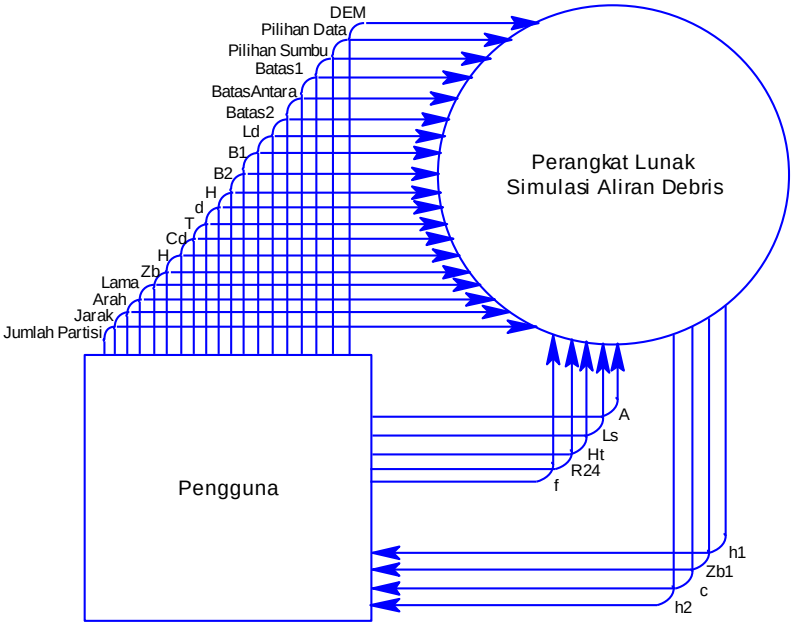
Penjelasan algoritma proses dekomposisi data dalam simulasi aliran debris satu dan dua dimensi pada Gambar 4.10 adalah sebagai berikut:

1. Buka data-data simulasi yang telah dihasilkan pada tahap sebelumnya (proses pengolahan data simulasi).
2. Tentukan batas awal dan akhir partisi pertama data ketinggian dua dimensi.
3. Setelah mengetahui batas awal dan batas akhir partisi data ketinggian dua dimensi, dapatkan partisi pertama data ketinggian dua dimensi dengan mengambil data ketinggian dua dimensi yang berada diantara kedua batas tersebut.
4. Lakukan proses simulasi aliran debris satu dimensi dengan menggunakan data ketinggian satu dimensi.
5. Jika proses simulasi aliran debris satu dimensi belum menggunakan semua data ketinggian satu dimensi (ditunjukkan dengan ketinggian aliran pada koordinat terakhir data ketinggian satu dimensi masih sama dengan nol) maka jalankan langkah 11. Namun, jika proses simulasi aliran debris satu dimensi telah menggunakan semua data ketinggian satu dimensi (ditunjukkan dengan ketinggian aliran pada koordinat terakhir data ketinggian satu dimensi tidak sama dengan nol) maka jalankan langkah 6.
6. Jika proses simulasi aliran debris dua dimensi belum berjalan atau proses simulasi aliran debris dua dimensi telah berjalan namun belum mencapai batas akhir partisi data ketinggian dua dimensi maka jalankan langkah 10. Namun, jika proses simulasi aliran debris dua dimensi telah berjalan dan telah mencapai batas akhir partisi data ketinggian dua dimensi maka jalankan langkah 7.
7. Tentukan batas awal dan akhir partisi data ketinggian dua dimensi selanjutnya.

8. Setelah mengetahui batas awal dan batas akhir partisi data ketinggian dua dimensi, dapatkan partisi selanjutnya data ketinggian dua dimensi dengan mengambil data ketinggian dua dimensi yang berada diantara kedua batas tersebut.
9. Dapatkan partisi data ketinggian dua dimensi saat ini dengan menggabungkan data ketinggian yang telah digunakan pada simulasi sebelumnya dengan partisi data ketinggian yang baru diperoleh pada langkah 8.
10. Jalankan proses simulasi aliran debris dua dimensi dengan menggunakan partisi data ketinggian dua dimensi saat ini.
11. Jika waktu aliran debris yang telah ditentukan sebelumnya telah habis maka proses dekomposisi data pada simulasi aliran debris satu dan dua dimensi telah selesai. Namun, jika waktu aliran debris masih berjalan maka jalankan langkah 4.

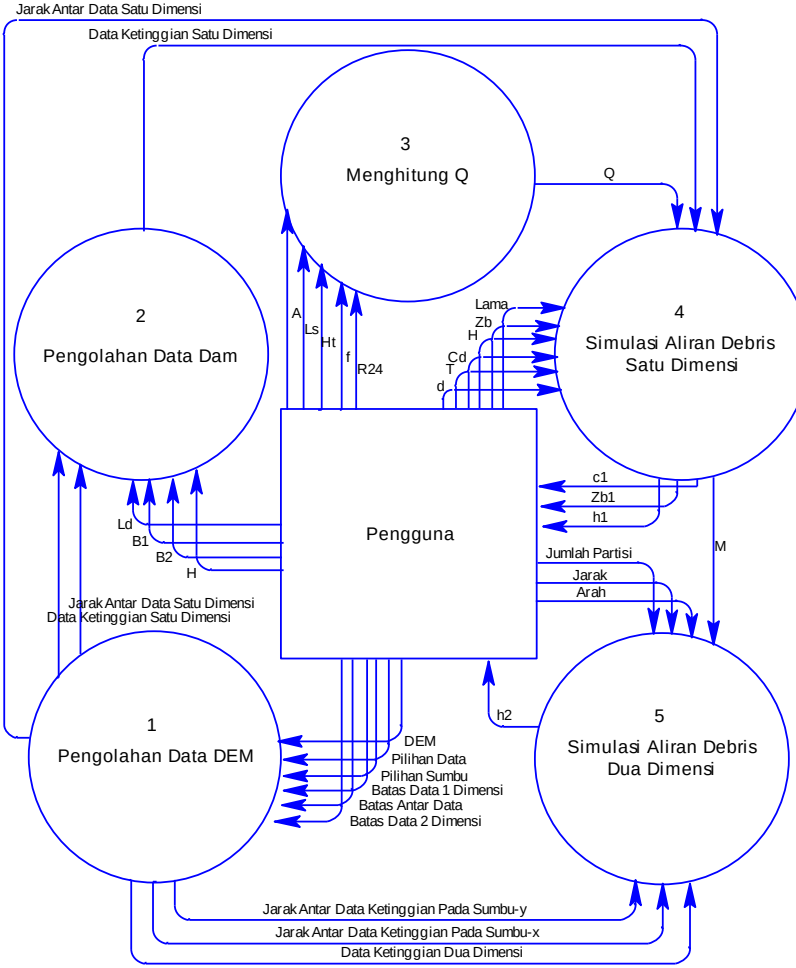
4. Data Flow Diagram (DFD)

Pemodelan data dalam simulasi aliran debris satu dan dua dimensi direpresntasikan dengan *Data Flow Diagram* (DFD) level 0 seperti terlihat pada Gambar 4.11. *Entit*y dalam sistem ini adalah pengguna. Pengguna akan memberikan masukan berupa DEM, data dam, parameter-parameter, lama simulasi, sumber aliran debris, jumlah partisi, yang kemudian akan diproses oleh perangkat lunak simulasi aliran debris dan menghasilkan keluaran berupa perubahan ketinggian lereng dan permukaan air yang kemudian ditampilkan sebagai model penyebaran aliran debris satu dan dua dimensi.



Gambar 4.11 DFD level 0 Simulasi Aliran Debris

Sistem simulasi aliran debris satu dan dua dimensi memiliki lima tahap yaitu pengolahan data DEM, pengolahan data dam, perhitungan debit, simulasi aliran debris satu dimensi dan simulasi aliran debris dua dimensi yang dinyatakan dengan DFD level 1 pada Gambar 4.12.



Gambar 4.12 DFD level 1 Simulasi Aliran Debris

Penjelasan setiap proses DFD level 1 Simulasi Aliran Debris pada Gambar 4.12 adalah sebagai berikut:

1. Proses Pengolahan Data DEM
   a. Uraian
      Dalam proses ini data DEM yang telah diperoleh kemudian diolah menjadi data ketinggian satu dan dua dimensi.
   b. Skenario *Input*
      Proses Pengolahan DEM menerima *input* utama berupa data DEM yang berektensi .xyz. Selain itu juga menerima *input* tambahan berupa pilihan data, pilihan sumbu, batas data satu dimensi, batas data dua dimensi dan batas antara data satu dimensi dan dua dimensi untuk mengolah data DEM.
   c. Syarat (Kondisi Awal)
      Data DEM berektensi .xyz tersedia.
   d. Hasil
      Data ketinggian satu dimensi, data ketinggian dua dimensi, jarak antar data ketiggian satu dimensi, jarak antar data ketinggian pada sumbu-x dan jarak antar data ketinggian pada sumbu-y.
2. Proses Pengolahan Data Dam
   a. Uraian
      Dalam proses ini data dam digunakan untuk menambahkan kontur data ketinggian satu dimensi.
   b. Skenario *Input*
      Proses menerima *input* dari pengguna yaitu data dam berupa nilai Ld, $B1$, $B2$, dan Hd. Selain itu proses ini juga menerima *input* berupa data ketinggian satu

dimensi dan jarak antar data ketinggian satu dimensi dari hasil proses pengolahan data DEM.

c. Syarat (Kondisi Awal)
Data ketinggian satu dimensi hasil proses pengolahan data DEM tersedia.

d. Hasil
Data ketinggian satu dimensi yang telah dilengkapi kondisi lerengnya dengan menggunakan data dam.

3. Proses Perhitungan Debit
   a. Uraian
   Dalam proses ini dilakukan perhitungan nilai debit dengan menggunakan metode rasional.
   b. Skenario *Input*
   Proses menerima *input* yaitu A, Ls, Ht, $R24$ dan f.
   c. Syarat (Kondisi Awal)
   A, Ls, Ht, $R24$ dan f tersedia.
   d. Hasil
   Nilai debit (Q).
4. Proses Simulasi Aliran Debris Satu Dimensi
   a. Uraian
   Dalam proses ini dilakukan proses simulasi aliran debris satu dimensi dengan menggunakan data-data yang telah dihasilkan dari proses-proses sebelumnya.
   b. Skenario *Input*
   Proses menerima *input* dari pengguna berupa d, T, Cd, H, Zb, dan lama simulasi. Selain itu, proses ini juga menerima *input* berupa nilai debit dari proses perhitungan nilai debit, data ketinggian satu dimensi dari proses pengolahan Dam dan jarak antar data satu dimensi dari proses pengolahan data DEM.

c. Syarat (Kondisi Awal)
   Data ketinggian satu dimensi, jarak antar data satu dimensi dan Q tersedia.
d. Hasil
   Konsentrasi sedimen (c), data ketinggian satu dimensi ($Zb1$) dan data kedalaman aliran satu dimensi (h1) yang ditampilakan sebagai gambaran aliran debris satu dimensi.

5. Proses Simulasi Aliran Debris Dua Dimensi
   a. Uraian
      Dalam proses ini dilakukan proses simulasi aliran debris dua dimensi dengan menggunakan data-data yang telah diolah pada proses-proses sebelumnya.
   b. Skenario *Input*
      Proses ini menerima *input* dari pengguna berupa jumlah partisi, jarak dan arah. Proses ini juga menerima nilai M dari proses simulasi aliran debris satu dimensi dan menerima data ketinggian dua dimensi, jarak antar data ketinggian pada sumu-x dan sumbu-y dari proses pengolahan data DEM.
   c. Syarat (Kondisi Awal)
      Data ketinggian dua dimensi, jarak antar data ketinggian pada sumbu-x dan sumbu-y dan M tersedia.
   d. Hasil
      Kedalaman aliran (h) yang ditampilakan sebagai gambaran aliran debris dua dimensi.

4.3 Hasil Implementasi

Perancangan program yang telah dibangun selanjutnya diimplementasikan pada bahasa pemrograman dengan menggunakan *software* Matlab R2009a. Pembahasan dalam implementasi sistem meliputi implementasi antarmuka *(interface)* sistem, implementasi tahap pengolahan data DEM, tahap pengolahan data simulasi dan tahap simulasi aliran.

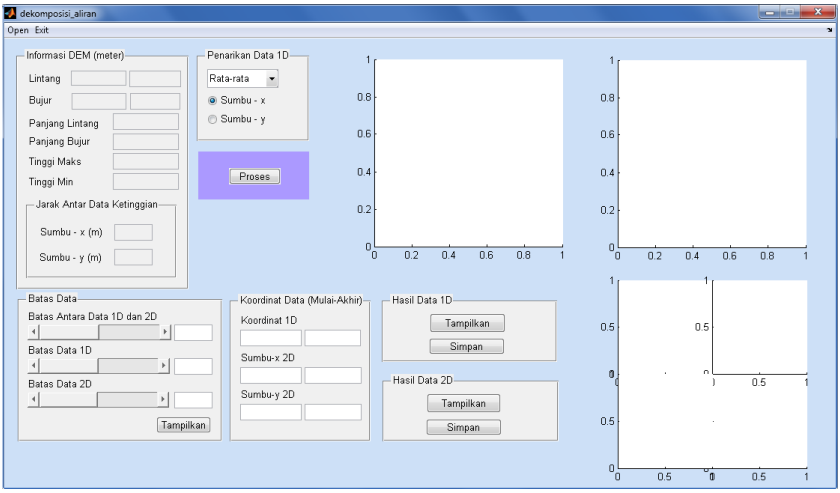
4.3.1 Implementasi Antarmuka

Pada Tugas Akhir ini, antarmuka sistem dibangun dengan menggunakan form dan kontrol yang terdapat pada Matlab R2009a. Adapun antarmuka-antarmuka yang diimplementasikan untuk menunjang penelitian Tugas Akhir ini adalah sebagai berikut.

1. Antarmuka Pengolahan Data DEM

Antarmuka pengolahan data DEM digunakan untuk mengolah DEM menjadi data ketinggian dalam bentuk matriks satu dimensi dan dua dimensi yang mudah diolah dengan menggunakan Matlab. Halaman antarmuka pengolahan data DEM dibuat dengan bentuk yang sederhana. Hasil implementasi antarmuka pengolahan data DEM dapat dilihat pada Gambar 4.13. Pada antarmuka ini terdapat *title bar* yang memiliki tombol *open* yang digunakan untuk membuka data DEM. Data DEM yang dapat dibuka adalah data DEM dengan format .xyz.

Panel informasi DEM digunakan untuk menampilkan informasi mengenai data DEM yang telah dibuka. Informasi tersebut meliputi koordinat garis lintang dan garis bujur, panjang garis lintang dan garis bujur, data ketinggian maksimum dan minimum dalam DEM, dan jarak antara data ketinggian satu dengan yang lainnya bedasarkan sumbu-x dan sumbu-y. Informasi-informasi ini dibutuhkan dalam pengambilan data satu dimensi dan data dua dimensi serta untuk membantu dalam menentukan batas data.



Gambar 4.13 Antarmuka Pengolahan Data DEM

Panel penarikan data satu dimensi digunakan untuk menentukan cara pemrosesan terhadap data DEM menjadi data ketinggian dalam bentuk matriks satu dimensi. Terdapat dua bagian pada *panel* ini, yaitu *radio button* dan menu *pop-up*. *Radio button* digunakan untuk menentukan penarikan data satu dimensi berdasarkan sumbu-x atau sumbu-y. Menu *pop-up* digunakan untuk menentukan penarikan data satu dimensi berdasarkan tiga pilihan yaitu nilai rata-rata, minimum, atau maksimum. Tombol proses digunakan untuk melakukan pemrosesan terhadap data DEM menjadi data ketinggian dalam bentuk matriks satu dimensi dan dua dimensi.

Pada *panel* batas data terdapat tiga buah *slider* yang digunakan untuk menentukan batas data satu dimensi dan data dua dimensi yang akan digunakan dalam proses simulasi. tiga buah *slider* tersebut adalah :

a. *Slider* batas antara data 1D dan 2D, merupakan *slider* yang digunakan untuk menentukan letak batas antara data ketinggian satu dimensi dan data ketinggian dua dimensi.

b. *Slider* data 1D, merupakan *slider* yang digunakan untuk menentukan letak batas data satu dimensi. *Slider* data 1D bersamaan dengan *slider* batas antar data 1D dan 2D digunakan untuk menentukan batas kanan dan kiri data satu dimensi.
c. *Slider* data 2D, merupakan *slider* yang digunakan untuk menentukan letak batas data ketinggian dua dimensi. *Slider* data 2D dan *slider* batas antara data 1D dan 2D merupakan slider yang digunakan untuk menentukan batas kanan dan kiri data ketinggian dua dimensi. Batas kanan dan kiri tersebut merupakan batas pada sumbu-x atau sumbu-y, yang telah ditentukan dari pilihan pada *radio button* dalam *panel* penarikan data satu dimensi.

Batas yang telah dipilih dengan menggunakan *slider* langsung ditampilkan. Selain tiga *slider* tersebut juga terdapat tiga buah kolom dengan kegunaan yang sama dengan slider dengan sebuah tombol (tampilkan) untuk menampilkan batas yang telah dimasukkan pada ketiga kolom tersebut.

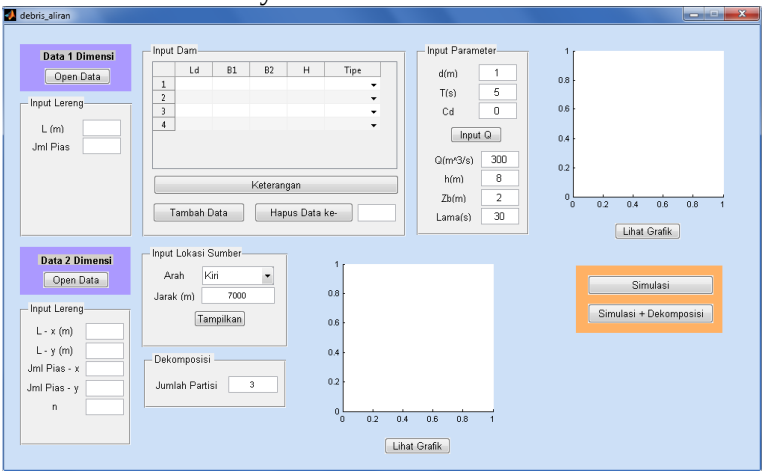
Pada panel koordinat data terdapat informasi mengenai koordinat data satu dan dua dimensi yang telah dipilih dengan menggunakan *slider*. Koordinat satu dimensi berisikan koordinat awal dan akhir pada data satu dimensi yang dipilih. Koordinat dua dimensi sumbu-x berisikan koordinat awal dan akhir pada sumbu-x, dan koordinat dua dimensi sumbu-y berisikan koordinat awal dan akhir pada sumbu-y.

Pada *panel* hasil data satu dimensi terdapat dua tombol, yaitu tombol tampilkan dan tombol simpan. Tombol tampilkan digunakan untuk menampilkan data satu dimensi yang telah ditentukan batas datanya sedangkan tombol simpan digunakan untuk menyimpan data satu dimensi dalam format .x1d. Pada *panel* hasil data dua dimensi terdapat dua tombol, yaitu tombol tampilkan dan tombol simpan. Tombol tampilkan digunakan untuk menampilkan data dua dimensi yang telah ditentukan batas datanya sedangkan tombol simpan digunakan untuk menyimpan data dua dimensi dalam format .x2d.

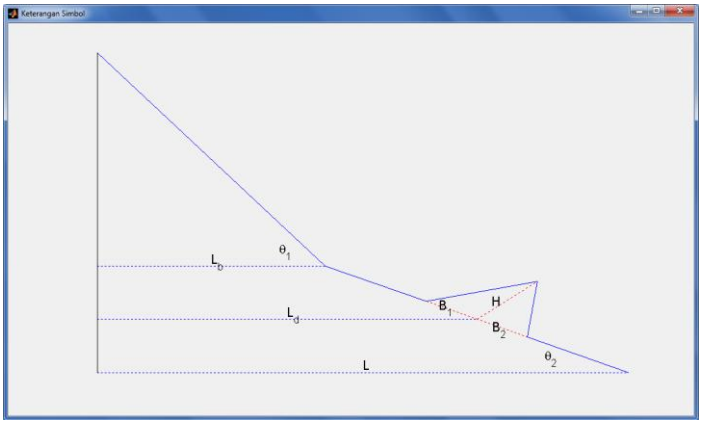
2. Antarmuka Pengolahan Data Simulasi

Antarmuka pengolahan data simulasi digunakan untuk memasukkan data-data yang dibutuhkan dalam simulasi. Hasil implementasi antarmuka pengolahan data simulasi dapat dilihat pada Gambar 4.14. Pada antarmuka pengolahan data simulasi terdapat dua tombol *open* yang masing-masing digunakan untuk membuka data ketinggian satu dan dua dimensi. Selain itu terdapat dua *panel* informasi lereng, dengan masing-masing panel digunakan untuk menampilkan informasi dari data ketinggian satu dan dua dimensi.

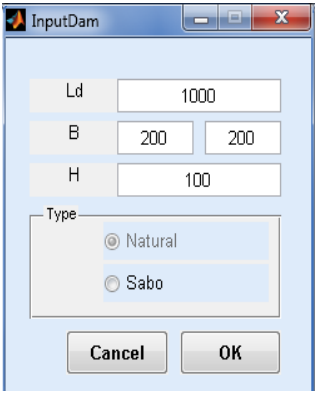
Panel input dam digunakan untuk memasukkan data-data dam. *Panel* ini memiliki tiga tombol yaitu tombol keterangan, tombol tambah data, dan tombol hapus data. Tombol keterangan digunakan untuk menampilkan keterangan simbol pada desain dam di lereng gunung. Keterangan simbol dam dapat ditunjukkan pada Gambar 4.15. Tombol tambah data digunakan untuk memangil antarmuka *input* dam, yang ditunjukkan pada Gambar 4.16. Pada antarmuka *input* dam terdapat 4 masukkan parameter yang harus dimasukkan yaitu $\boldsymbol{Ld}$, $\boldsymbol{B1}$, $\boldsymbol{B2}$, dan $\boldsymbol{H}$. Tombol hapus data digunakan untuk menghapus data masukan dam yang telah dimasukkan sebelumnya.



Gambar 4.14 Antarmuka Pengolahan Data Simulasi



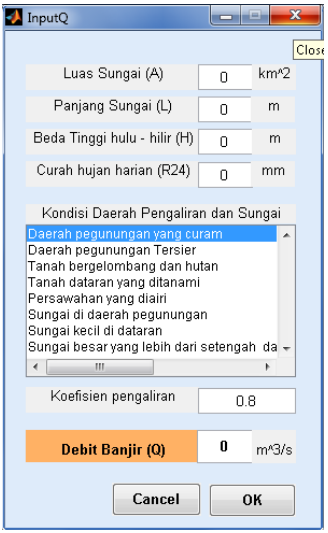
Gambar 4.15 Keterangan Simbol



Gambar 4.16 Antarmuka *Input* Dam

Panel input parameter digunakan untuk memasukkan paramater-parameter yang dibutuhkan dalam simulasi meliputi $d(m)$, $T(s)$, Cd, $Q(m^3/s)$, $h(m)$, $Z_b(m)$ dan lama simulasi(s). Selain itu terdapat tombol *input* Q yang digunakan untuk menampilkan antarmuka *input* Q. Antarmuka *input* Q digunakan untuk menghitung debit. Terdapat tiga buah masukan yaitu luas sungai(A), panjang sungai(L), beda ketinggian hulu-hilir(H) dan

curah hujan harian ($R24$). Selain itu juga dipilih kondisi daerah pengaliran dan sungai untuk mendapatkan koefisien pengaliran. Nilai debit akan ditampilkan setelah semua masukkan diisi. Terdapat dua tombol yaitu tombol *cancel* untuk membatalkan proses perhitungan debit dan kembali ke antarmuka *input* simulasi serta tombol oke untuk menggunakan nilai debit yang diperoleh sebagai masukkan pada *panel input* parameter. Antarmuka *input* Q dapat dilihat pada Gambar 4.17 berikut.



Gambar 4.17 Antarmuka *Input* Debit

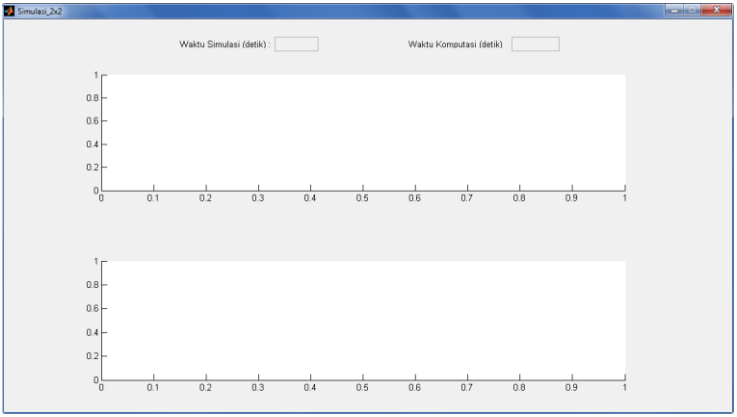
Pada panel *input* lokasi sumber terdapat menu *popup* arah yang digunakan untuk menentukan arah dari aliran debris. Pada menu *pupup* arah ini terdapat empat pilihan yaitu kiri, kanan, atas dan bawah. Kolom jarak digunakan untuk memasukkan jarak dari lokasi sumber. Apabila memilih arah kiri atau kanan maka jarak yang dimasukkan merupakan jarak lokasi sumber aliran debris pada peta dari sumbu-x, sedangkan jika memilih arah atas atau bawah maka jarak yang dimasukkan adalah jarak lokasi sumber aliran debris pada peta dari sumbu-y. Selain itu juga terdapat

tombol tampilkan untuk menampilkan titik lokasi sumber aliran debris berdasarkan arah dan jarak yang t elah dipilih sebellumnya.

Pada *panel* dekomposisi terdapat kolom jumlah partisi yang digunakan untuk menentukan banyaknya partisi yang digunakan ketika menjalankan simulasi dengan menggunakan dekomposisi. Pada antarmuka *input* simulasi terdapat dua buah tombol lihat grafik yang masing-masing digunakan untuk melihat grafik data ketinggian satu dimensi (tampak samping) dan dua dimensi (tampak atas) secara penuh. Selain itu terdapat tombol simulasi untuk melakukan simulasi tanpa menjalankan dekomposisi data serta tombol simulasi + dekomposisi untuk melakukan simulasi dengan menjalankan dekomposisi didalamnya selama simulasi berlangsung.

3. Antarmuka Simulasi Aliran

Antarmuka simulasi aliran digunakan untuk menampilkan hasil dari proses simulasi. Terdapat dua kolom yaitu kolom waktu simulasi yang menunjukkan waktu yang dibutuhkan dalam simulasi aliran debris dan kolom waktu komputasi merupakan waktu yang dibutuhkan komputer untuk melakukan proses simulasi. Antarmuka simulasi dapat dilihat pada Gambar 4.18 berikut.



Gambar 4.18 Antarmuka Simulasi

4.3.2 Implementasi Proses Pengolahan Data DEM

Proses pengolahan data DEM adalah proses mengolah data DEM menjadi data ketinggian dalam bentuk matriks satu dimensi dan dua dimensi untuk menyederhanakan struktur data sehingga data tersebut lebih mudah ditampilkan dalam simulasi aliran debris satu dan dua dimensi.

Data ketinggian satu dimensi diperoleh dengan menghitung nilai rata-rata, minimum atau maksimum dari data DEM berdasarkan sumbu-x atau sumbu-y. Kode pemrograman untuk mendapatkan data ketinggian dalam bentuk matriks satu dimensi sebagai berikut.

```
DEM = sortrows(DEM,axisBasedData);
a=1;%start
zb_init = [];
for x = 2:length(DEM(:,axisBasedData))
    if  DEM(x,axisBasedData)==DEM(x-
1,axisBasedData)
        b=x; %end
    else
        if dataSelect == 1
            tempZb = mean(DEM(a:b,3));
        elseif dataSelect == 2
            tempZb = min(DEM(a:b,3));
        else
            tempZb = max(DEM(a:b,3));
        end
        zb_init = cat(1,zb_init,tempZb);
        a=x;%start
    end
end
if DEM(x,axisBasedData)==DEM(x-1,axisBasedData)
    if dataSelect == 1
        tempZb = mean(DEM(a:b,3));
    elseif dataSelect == 2
        tempZb = min(DEM(a:b,3));
    else
        tempZb = max(DEM(a:b,3));
```

```
    end
    zb_init = cat(1,zb_init,tempZb);
else
    tempZb =
DEM(length(DEM(:,axisBasedData)),3);
    zb_init = cat(1,zb_init,tempZb);
end
```

Data ketinggian dalam bentuk matriks dua dimensi diperoleh dengan meletakkan nilai-nilai data ketinggian DEM kedalam sebuah matriks dua dimensi yang berukuran (banyak data pada sumbu-x × banyak data pada sumbu-y) DEM. Kode pemrograman dalam mendapatkan data ketinggian dalam bentuk matriks dua dimensi adalah sebagai berikut.

```
num_sectionx =
(max(DEM(:,1))-min(DEM(:,1)))/resolutionx;

num_sectiony =
(max(DEM(:,2))-min(DEM(:,2)))/resolutiony;

DEM(:,1) = (DEM(:,1)-min(DEM(:,1)))/resolutionx;
DEM(:,2) = (DEM(:,2)-min(DEM(:,2)))/resolutiony;

zb_2 = zeros([num_sectionx num_sectiony]);
for x=1:length(DEM)
    zb_2(DEM(x,1)+1,DEM(x,2)+1)=DEM(x,3);
end
```

Berdasarkan interpretasi fisik data ketinggian satu dan dua dimnesi harus dipotong dimana data satu dimensi berada pada lereng atas dan data dua dimensi pada lereng bawah. Pada implementasi program, pemotongan data tidak dilaksanakan pada saat pengguna mengubah batas data satu dimensi dan dua dimensi, namun pada saat penyimpanan data.

Data ketinggian satu dan dua dimensi yang disimpan pada proses penyimpanan data adalah data ketinggian yang berada pada batas-batas yang telah ditentukan. Dengan cara ini, program

tidak melakukan pemotongan dan penyimpanan data setiap kali pengguna melakukan perubahan batas namun cukup pada saat pengguna melakukan penyimpanan data saja. Kode pemrograman dalam proses penyimpanan data satu dan dua dimensi dalam fungsi berikut.

```
function btnSimpan1D_Callback(hObject,
eventdata, handles)
function btnSimpan2D_Callback(hObject,
eventdata, handles)
```

Data ketinggian satu dimensi, jarak antar data ketinggian satu dimensi, dan koordinat terkecil data satu dimensi disimpan dalam sebuah file dengan ekstensi .x1d. Sedangkan data ketinggian dua dimensi, jarak antar data ketinggian dua dimensi pada sumbu-x dan sumbu-y, serta koordinat terkecil data dua dimensi pada garis lintang dan garis bujur disimpan dalam file dengan ekstensi .x2d. Kode pemrograman pengolahan data DEM selengkapnya dapat dilihat pada Lampiran A1.

4.3.3 Implementasi Proses Pengolahan Data Simulasi

Proses pengolahan data simulasi bertujuan untuk mengumpulkan data-data yang dibutuhkan dalam proses simulasi dan mengolahnya terlebih dahulu sebelum digunakan. Data-data tersebut adalah data ketinggian satu dan dua dimensi, data dam sabo, parameter-parameter simulasi, jumlah partisi, dan lokasi sumber aliran debris.

Data simulasi (beserta nama variabel) yang dimasukkan pada tahap ini ditunjukkan sebagai berikut.

1. Data ketinggian satu dimensi, terdiri atas matriks data ketinggian satu dimensi (zb_init), koordinat terkecil data satu dimensi(min1) dan jarak antar data ketinggian (resolution).
2. Data ketinggian dua dimensi, terdiri atas matriks data ketinggian dua dimensi (zb_2), koordinat terkecil pada sumbu-x (minx), koordinat terkecil pada sumbu-y (miny),

jarak antar data ketinggian berdasarkan sumbu-x (resolutionx) dan jarak antar data ketinggian berdasarkan sumbu-y (resolutiony).

3. Data sabo dam, terdiri atas jarak antara titik sumber aliran dan center dam (Ld), jarak antara center dam dengan titik atas dam (B1), jarak antara center dam dengan titik bawah dam (B2) dan ketinggian dam (H).
4. Data parameter, terdiri atas diameter sedimen (d), durasi waktu (T), konsentrasi sedimen (Cd), debit (Q), kedalaman aliran saat puncak (h), tinggi lereng (Zb), dan lama simulasi.
5. Data lokasi sumber yang terdiri atas arah dan jarak.
6. Jumlah partisi.

Sabo dam adalah bangunan yang digunakan untuk menahan aliran debris. Dalam implementasi pemrograman, data sabo dam ini digunakan untuk mengubah lereng kali putih. Proses pengolahan data sabo dam diimplementasikan kedalam fungsi berikut ini.

```
function retVal = InsertSaboDam(hObject,
handles, Ld, B1, B2, H)
```

Nilai debit dalam parameter simulasi dapat dimasukkan secara langsung maupun melalui perhitungan debit terlebih dahulu dengan cara memasukkan nilai luas sungai (A), panjang sungai(L), beda tinggi hulu hilir (Ht), curah hujan harian (R24) dan koefisien pengaliran yang didasarkan pada kondisi daerah pengaliran dan sungai. Kode pemrograman dalam mendapatkan nilai debit adalah sebagai berikut.

```
W = 72*((H/L)^0.6);
t = L/W;
r = (R24/24) * ((24/t)^(2/3));
Q = (1.36)*f*r*A;
```

Data lokasi sumber aliran debris terdiri atas arah aliran dan jarak. Jarak yang dimaksud adalah jarak koordinat tersebut dari sumbu-x atau sumbu-y. Jumlah partisi digunakan untuk menentukan banyaknya partisi data dalam proses dekomposisi data. Kode pemrograman pengolahan data simulasi selengkapnya dapat dilihat pada Lampiran A2.

4.3.4 Implementasi Proses Simulasi Aliran

Proses simulasi aliran debris bertujuan untuk menggambarkan aliran debris di Kali Putih Gunung Merapi. Didalam simulasi ini peneliti melakukan dekomposisi data untuk memgurangi waktu komputasi yang dibutuhkan. Proses ini diawali dengan mendapatkan partisi pertama data ketinggian dua dimensi. Kode pemrograman dalam mendapatkan partisi pertama data ketinggian dua dimensi adalah sebagai berikut.

```
if arah == 1 %kiri
    if sisa_part > 0
        batasx_end = round( (jml_part-
sisa_part)/jml_part
        *(num_sectionx_all-1));
    else
        batasx_end = num_sectionx_all-1;
    end
    batasx_start = 1;
    batasy_start = 1;
    batasy_end = num_sectiony-1;
elseif arah == 2 %kanan
    if sisa_part > 0
        batasx_start = round( sisa_part/jml_part
        *(num_sectionx_all-1));
    else
        batasx_start = 1;
    end
    batasx_end = num_sectionx-1;
    batasy_start = 1;
    batasy_end = num_sectiony-1;
elseif arah == 3 %atas
```

```
    if sisa_part > 0
        batasy_start = round( sisa_part/jml_part
        *(num_sectiony_all-1));
    else
        batasy_start = 1;
    end
    batasx_start = 1;
    batasx_end = num_sectionx-1;

batasy_end = num_sectiony-1;
elseif arah == 4 %bawah
    if sisa_part > 0
        batasy_end = round( (jml_part-
sisa_part)/jml_part
        *(num_sectiony_all-1));
    else
        batasy_end = num_sectiony_all-1;
    end
    batasx_start = 1;
    batasx_end = num_sectionx-1;
    batasy_start = 1;
end

batasb  = batasx_start:batasx_end;
batask  = batasy_start:batasy_end;
batasb2 = batasx_start:batasx_end+1;
batask2 = batasy_start:batasy_end+1;

zb2_new = zb2(batasb,batask);
n_new = n(batasb,batask);
[num_sectionx_new num_sectiony_new] =
size(zb2_new);

Lx_new = (num_sectionx_new-1)*dx_2;
Ly_new = (num_sectiony_new-1)*dy_2;
num_sectionx_new = num_sectionx_new+1;
num_sectiony_new = num_sectiony_new+1;
% inisialisasi data 2d
M2_new =zeros(num_sectionx_new,num_sectiony_new-
1);
```

```
N2_new = zeros(num_sectionx_new -
1,num_sectiony_new);
h2_old_new = zeros(num_sectionx_new -
1,num_sectiony_new - 1);
h2_new_new = zeros(num_sectionx_new -
1,num_sectiony_new - 1);
```

Proses simulasi dilanjutkan dengan simulasi aliran debris satu dimensi. Setelah proses simulasi aliran debris satu dimensi, kemudian dilanjutkan dengan simulasi aliran debris dua dimensi. Simulasi aliran debris satu dimensi tetap berjalan ketika simulasi aliran debris dua dimensi berjalan.

Apabila waktu simulasi yang ditentukan belum usai, maka terdapat dua kemungkinan yaitu proses simulasi telah mencapai batas partisi data dua dimensi atau belum. Jika belum mencapai batas partisi maka simulasi aliran debris satu dan dua dimensi akan tetap dilanjutkan, namun jika simulasi aliran debris dua dimensi telah mencapai batas partisi maka data partisi sebelumnya digabungkan dengan data partisi selanjutnya.

Selanjutnya proses simulasi aliran debris dilanjutkan dengan menggunakan data partisi gabungan ini. Ketika waktu simulasi habis maka dapat terlihat hasil simulasi berupa gambaran aliran debris satu dan dimensi dalam waktu aliran yang telah ditentukan dan waktu komputasi yang diperlukan. Kode pemrograman proses simulasi selengkapnya dapat dilihat pada Lampiran A3.

BAB V
PENGUJIAN DAN PEMBAHASAN HASIL

Bab ini menjelaskan mengenai proses pengujian yang dilakukan terhadap program dekomposisi data dalam simulasi aliran debris satu dan dua dimensi. Hasil pengujian kemudian dibahas untuk mengetahui hasil kerja program secara keseluruhan dalam menjalankan fungsi yang diharapkan. Isi dari bab ini dimulai dengan mengenalkan lingkungan pengujian sistem yang digunakan. Selanjutnya dijelaskan mengenai hasil pengujian terhadap sistem yang telah diimplementasikan pada bab 4, yaitu pengujian tahap pengolahan DEM, tahap pengolahan data simulasi dan tahap dekomposisi data dalam simulasi aliran.

5.1 Lingkungan Pengujian Sistem

Lingkungan pengujian dari program dekomposisi data dalam simulasi aliran debris satu dan dua dimensi meliputi perangkat keras dan lunak komputer. Detail dari perangkat keras dan lunak yang digunakan dapat dilihat pada Tabel 5.1

Tabel 5.1 Lingkungan Pengujian Sistem

Perangkat Keras	Prosesor : Intel(R) Core(TM) i3 – 2350M CPU @2.3GHz
	Memory : 2 GB
Perangkat Lunak	Sistem Operasi : Windows 7 Ultimate 32-bit
	Tools : MATLAB 7.8.0 (R2009a)

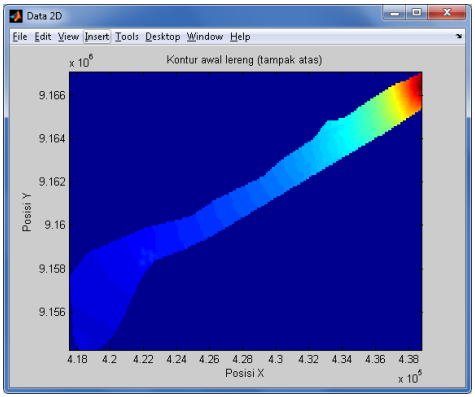
5.2 Pengujian Tahap Pengolahan DEM

Tujuan dari pengujian tahap pengolahan DEM adalah untuk mengetahui bahwa program telah berhasil mengolah DEM menjadi data ketinggian satu dan dua dimensi.

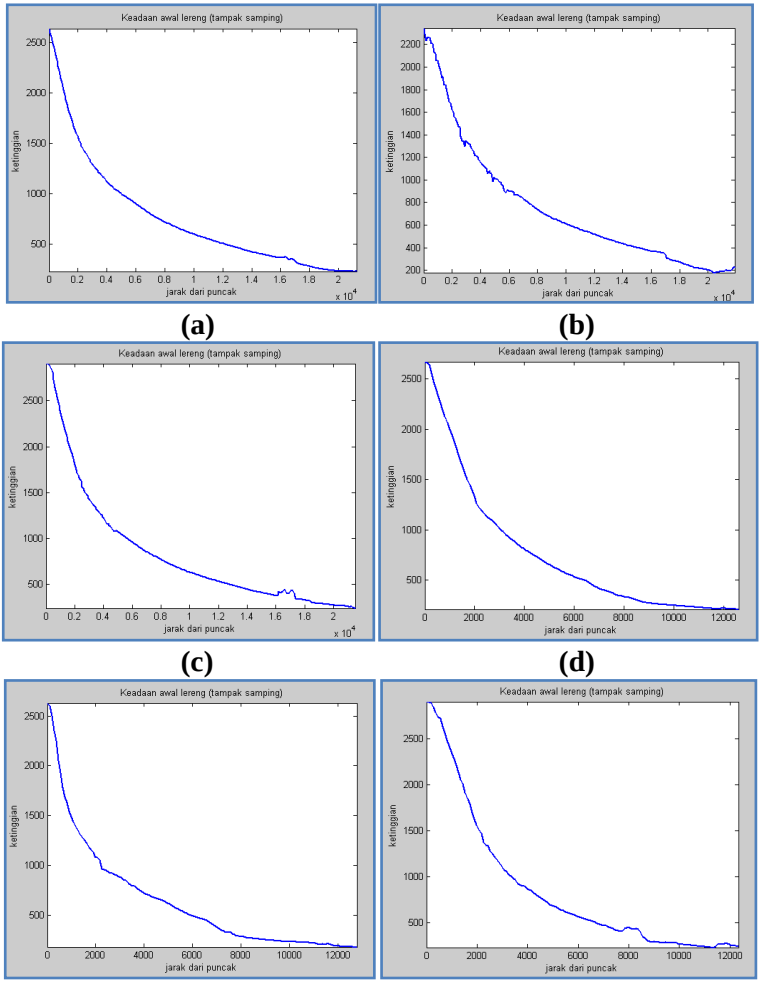
1. Pengujian penarikan data ketinggian satu dimensi, bertujuan untuk mengetahui bahwa program berhasil mengolah DEM menjadi data ketinggian satu dimensi berdasarkan nilai rata-rata, minimum atau maksimum pada

sumbu-x atau sumbu-y. Hasil pengujian tahap ini ditunjukkan pada Gambar 5.2.

2. Pengujian penarikan data ketinggian dua dimensi, bertujuan untuk mengetahui bahwa program berhasil mengolah DEM menjadi data ketinggian dua dimensi. Hasil pengujian tahap ini ditunjukkan pada Gambar 5.1.
3. Pengujian penyimpanan data berdasarkan batas, bertujuan untuk mengetahui bahwa program berhasil menyimpan data ketinggian satu dimensi dan dua dimensi pada batas yang telah ditentukan. Hasil pengujian tahap ini ditampilkan pada Tabel 5.2. Data ketinggian satu dimensi pada Tabel 5.2 merupakan hasil penarikan data ketinggian satu dimensi berdasarkan nilai rata-rata dan sumbu-x.

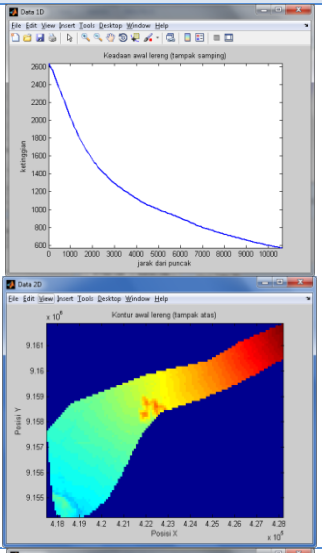
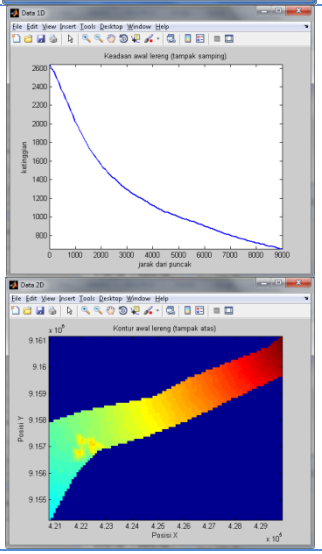


Gambar 5.1 Hasil Penarikan Data Ketinggian Dua Dimensi



Gambar 5.2 Hasil Penarikan Data Ketinggian Satu Dimensi Berdasarkan Ketentuan (a) Nilai Rata-Rata pada Sumbu-x, (b) Nilai Minimum pada Sumbu-x, (c) Nilai Maksimum pada Sumbu-x, (d) Nilai Rata-Rata pada Sumbu-y, (e) Nilai Minimum pada Sumbu-y, (f) Nilai Maksimum pada Sumbu-y,

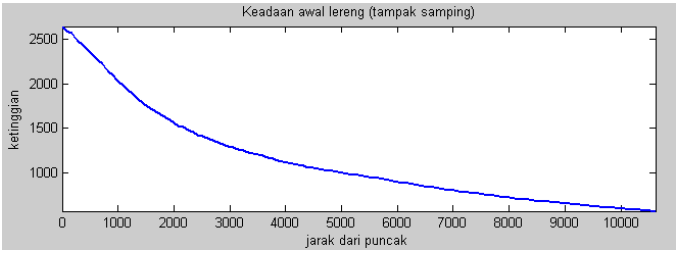
Tabel 5.2 Hasil Pemotongan Data Berdasarkan Batas

Batas	Hasil
Batas data 1D = 21240 Batas antara data 1D dan 2D = 10620 Batas data 2D = 0	
Batas data 1D = 21240 Batas antara data 1D dan 2D = 12240 Batas data 2D = 3240	

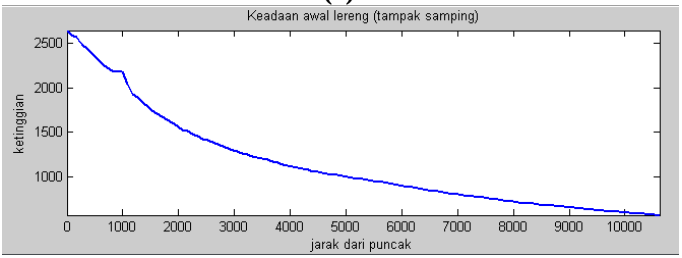
5.3 Pengujian Tahap Pengolahan Data Simulasi

Pengujian tahap pengolahan data simulasi bertujuan untuk mengetahui bahwa program berhasil mengolah data simulasi.

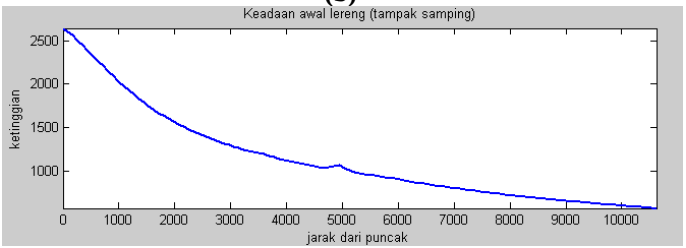
1. Pengujian tahap pengolahan data dam, bertujuan untuk mengetahui bahwa program berhasil mengolah data dam. hasil pengolahan ini kemudian digunakan untuk mengubah kondisi awal lereng (data ketinggian satu dimensi). Hasil pengujian tahap ini ditunjukkan pada Gambar 5.3.
2. Pengujian tahap perhitungan debit, bertujuan untuk mengetahui bahwa program berhasil menghitung nilai debit. Hasil pengujian tahap ini ditunjukkan pada Gambar 5.4.
3. Pengujian tahap memasukkan lokasi sumber, bertujuan untuk mengetahui bahwa program berhasil memasukkan lokasi sumber aliran debris. Lokasi sumber yang dimaksudkan adalah lokasi sumber aliran debris pada kontur lereng (data ketinggian dua dimensi). Hasil pengujian pada tahap ini ditunjukkan pada Tabel 5.3.



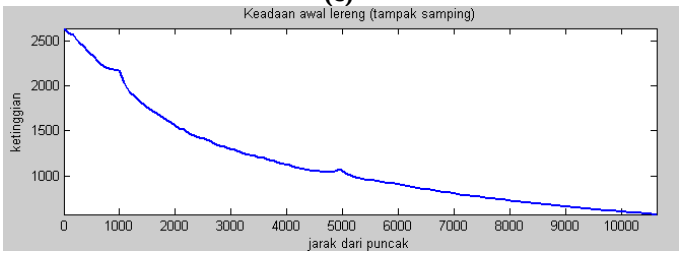
(a)



(b)

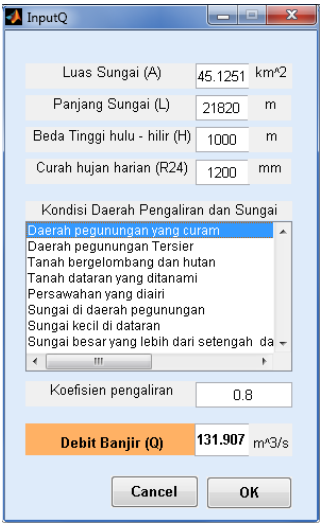


(c)



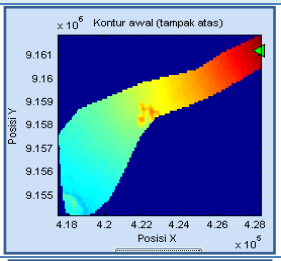
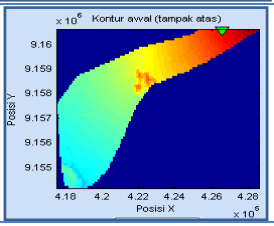
(d)

Gambar 5.3 Hasil Pengolahan Data Dam(a) Tanpa Dam, (b) Ld = 1000, B1 = 200, B2 = 200, H = 100, (c)Ld = 5000, B1=100, B2 = 100, H = 50 (d)Gabungan Kondisi (b) dan (c)



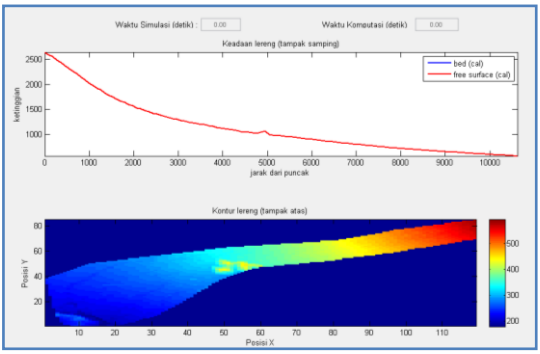
Gambar 5.4 Hasil Perhitungan Debit

Tabel 5.3 Hasil pengolahan lokasi sumber

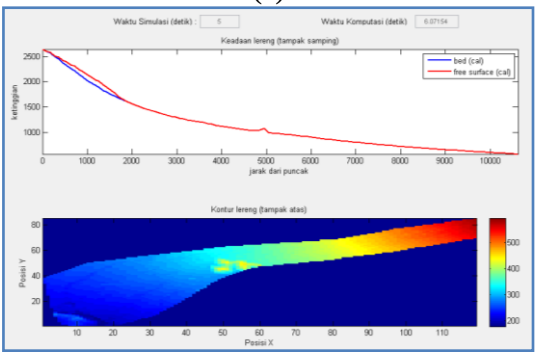
Lokasi Sumber	Hasil
Arah = kanan Jarak = 7000 meter Koordinat data = (417547.199 – 428167.199, 9154193.433 – 9175433.433)	
Arah = atas Jarak = 9000 meter Koordinat data = (417547.199 – 430147.199, 9154193.433 – 9160493.433)	

5.4 Pengujian Tahap Simulasi Aliran

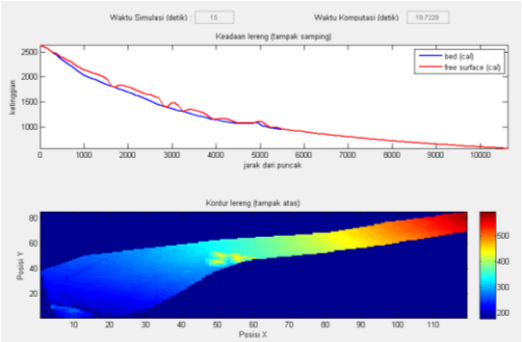
Pengujian tahap simulasi aliran bertujuan untuk mengetahui bahwa program berhasil mendapatkan gambaran aliran debris satu dimensi dan dua dimensi serta kecepatan komputasi yang diperoleh setelah menggunakan dekomposisi data di dalam proses simulasi aliran. Beberapa hasil simulasi aliran debris ditunjukkan pada Gambar 5.5 dan perbandingan waktu komputasi simulasi penyebaran model aliran debris yang menggunakan dekomposisi data dengan jumlah partisi 119 dan pengujian tanpa dekomposisi data disajikan dalam Tabel 5.4.



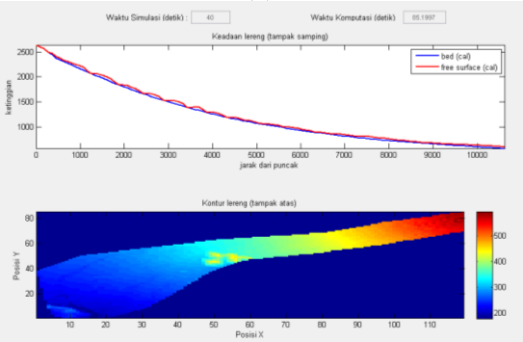
(a)



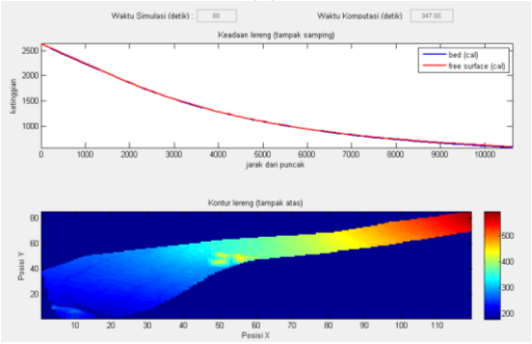
(b)



(c)



(d)



(e)

Gambar 5.5 Hasil Simulasi Aliran Debris (a) 0 detik, (b) 5 detik, (c) 15 detik, (d) 40 detik, (e) 80 detik

Tabel 5.4 Perbandingan Hasil Pengujian Simulasi Penyebaran Aliran Debris Satu Dimensi dan Dua Dimensi dengan Menggunakan Dekomposisi Data (Jumlah Partisi adalah 119) dan Tanpa Melakukan Dekomposisi Data.

Waktu Aliran (detik)	Waktu Komputasi (detik)	
	Menggunakan Dekomposisi Data	Tanpa Dekomposisi Data
10	14,1575	55,7127
20	28,0669	165,615
30	46,7515	250,505
40	96,288	342,181
50	160,706	404.221
60	224,147	623.19
70	296,040	702.167
80	390,982	781.208
90	478,679	819.332

Dari hasil pada Tabel 5.4, dekomposisi data dalam proses simulasi aliran debris satu dan dua dimensi berhasil memperkecil waktu proses simulasi. Hal ini terjadi karena dalam proses simulasi program tidak melakukan penghitungan terhadap semua data ketinggian secara langsung namun melakukan penghitungan terhadap bagian-bagian data. Jumlah data yang digunakan bertambah setiap kali perhitungan telah mencapai tepi matriks sehingga penurunan waktu komputasi terlihat semakin lama semakin mengecil.

5.5 Pembahasan Hasil Pengujian

Pembahasan hasil pengujian difokuskan pada hasil pengujian simulasi penyebaran aliran debris satu dan dua dimensi. Pembahasan hasil pengujian ini digunakan untuk mengetahui penurunan waktu komputasi setelah melakukan dekomposisi data. Dilihat dari Tabel 5.4 program berhasil mengurangi waktu komputasi dengan baik. Kemudian dilakukan perhitungan tingkat

penurunan waktu komputasi program setelah menggunakan dekomposisi data. Perhitungan tingkat penurunan waktu komputasi ini berdasarkan pada:

$$PK = \frac{WTD - WD}{WTD} \times 100\%$$

dengan,
PK = Prosentase penurunan waktu komputasi
WTD = Total Waktu Komputasi Tanpa Dekomposisi Data
WD = Total Waktu Komputasi Dengan Dekomposisi Data

Sehingga tingkat penurunan waktu komputasi program simulasi aliran debris satu dan dua dimensi dengan menggunakan dekomposisi data dapat dilihat pada Tabel 5.5 berikut ini.

Tabel 5.5 Prosentase Penurunan Waktu Komputasi Simulasi Penyebaran Aliran Debris Satu dan Dua Dimensi dengan Menggunakan Dekomposisi Data (Jumlah Partisi adalah 119) .

Waktu Komputasi (detik)		Prosentase Penurunan Waktu Komputasi
Menggunakan Dekomposisi Data	Tanpa Dekomposisi Data	
14,1575	55,7127	71.5884%
28,0669	165,615	83.0529%
46,7515	250,505	81.3371%
96,288	342,181	71.8605%
160,706	404.221	60.2430%
224,147	623.19	64.0323%
296,040	702.167	57.8391%
390,982	781.208	49.9516%
478,679	819.332	41.5769%

Dari data prosentase keberhasilan pada Tabel 5.5 program dapat menurunkan waktu komputasi dengan tingkat penurunan waktu komputasi mencapai 41.5769%.

“Halaman ini sengaja dikosongkan”

BAB VI
PENUTUP

Bab ini berisi kesimpulan pengujian dan hasil penelitian dekomposisi data dalam Tugas Akhir ini. Selain itu, juga diberikan saran yang dapat digunakan untuk mengembangkan penelitian.

6.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan diperoleh kesimpulan bahwa :

1. Dekomposisi data berhasil dijalankan dalam simulasi model penyebaran aliran debris satu dan dua dimensi dengan menggunakan data DEM.
2. Hasil simulasi menunjukkan bahwa dekomposisi data dalam Tugas Akhir ini berhasil menurunkan waktu komputasi sebesar 41.5769%.

6.2 Saran

Adapun saran dalam Tugas Akhir ini adalah sebagai berikut:

1. Dekomposisi data yang dilakukan pada Tugas Akhir ini dilakukan dangan melakukan partisi data berdasarkan sumbu-x atau sumbu-y, dalam pengembangannya sebaiknya pemotongan data berdasarkan sumbu-x dan sumbu-y.
2. Penelitian berikutnya diharapkan menggunakan data DEM dengan resolusi yang lebih kecil.

“Halaman ini sengaja dikosongkan”

DAFTAR PUSTAKA

[1] Adzkiya, D., Gazali, M., Sanjoyo, B.A., dan Trisunarno, L. 2008. "Diskritisasi Model Penyebaran Aliran Debris 2 Dimensi", Seminar Nasional Matematika IV, ITS, 13 Desember.

[2] Prahasta, Eddy. 2009. "Sistem Informasi Geografis : Konsep-konsep Dasar (Perspektif Geodesi & Geomatika)". Penerbit Informatika, Bandung

[3] Putro, Suyitno Hadi. "Dampak Bencana Aliran Lahar Dingin Gunung Merapi Pasca Erupsi di Kali Putih", Seminar Nasional : Pengembangan Kawasan Merapi : Aspek Kebencanaan dan Pengembangan Masyarakat Pasca Bencana.

[4] Sanjoyo, B.A., dan Adzkiya, D. 2010. "One and Two Dimentional Debris Flow Simulation Using Finite Difference Method", International Conference on Mathematical Application in Engineering (ICMAE'10), Kuala Lumpur, Malaysia, 3rd – 4th August.

[5] Sanjoyo, B.A., Adzkiya, D., dan Trisunarno, L. 2009. "Simulasi Penyebaran Aliran Debris 1 Dimensi dengan Metode Beda Hingga", Seminar Nasional Matematika 2009, Fakultas FMIPA Universitas Jember, 28 Februari.

[6] Sarkar, Vivek. 2008. "Decomposition Techniques for Parallel Alghoritms" diakses dari http://www.cs.rice.edu /~vs3/comp422/lecture-notes/comp422-lec4-s08-v1.pdf tanggal 6 Oktober 2014

[7] Subagio, Narieswari. 2010. "Investigasi Data Dem: Konstruksi Data dan Aplikasi Dalam Bidang Hidrologi Permukaan".

[8] Sulistiyono, B. 2013. "Teknik Sabo Cegah Aliran Debris" diakses dari http://www.ugm.ac.id /id/post/ page?id=5297 tanggal 5 Oktober 2014 pukul 11.17 WIB.

[9] Wibowo, Eko Y. 2014. "Digital Elevation Model (DEM)" diakses dari http://ekowibowo78.blogspot.com/2014/04/ digital-elevation-model-dem.html tanggal 6 Oktober 2014.

DAFTAR LAMPIRAN

“Halaman ini sengaja dikosongkan”

LAMPIRAN A

A.1 Kode Program Untuk Pengolahan DEM

1. Fungsi Open

```
function Open_Callback(hObject, eventdata,
handles)

[file,path] = uigetfile('*.xyz');
if (file == 0)
 return;
end
filename = strcat(path,file);

DEM = importdata(filename);
handles.DEM = DEM;

DEM = sortrows(DEM,1);
for i = 1:length(DEM)
    resx = DEM(i+1,1)-DEM(i,1);
    if resx ~= 0
        break;
    end
end

DEM = sortrows(DEM,2);
for i = 1:length(DEM)
    resy = DEM(i+1,2)-DEM(i,2);
    if resy ~= 0
        break;
    end
end
```

LAMPIRAN A (LANJUTAN)

```
%INFORMATION
set(handles.editLintang1,'String',
num2str(min(DEM(:,1))));
set(handles.editLintang2,'String',
num2str(max(DEM(:,1))));
set(handles.editBujur1,'String',
num2str(min(DEM(:,2))));
set(handles.editBujur2,'String',
num2str(max(DEM(:,2))));
set(handles.editPLintang,'String',
num2str(max(DEM(:,1))-min(DEM(:,1))));
set(handles.editPBujur,'String',
num2str(max(DEM(:,2))-min(DEM(:,2))));
set(handles.editTMaks,'String',
num2str(max(DEM(:,3))));
set(handles.editTMin,'String',
num2str(min(DEM(:,3))));
set(handles.editResX,'String', resx);
set(handles.editResY,'String', resy);

guidata(hObject, handles);
```

2. Fungsi Pilihan Sumbu

```
function uipanel16_SelectionChangeFcn
(hObject, eventdata, handles)

if hObject == handles.radiobuttonX
    axisBasedData = true;
else
    axisBasedData = false;
end
handles.axisBasedData = axisBasedData;
guidata(hObject, handles);
```

LAMPIRAN A (LANJUTAN)

3. Fungsi Proses

```
function btnProses_Callback(hObject,
eventdata, handles)

% Ambil DATA
DEM = handles.DEM;
resolutionx =
str2num(get(handles.editResX,'String'));
resolutiony =
str2num(get(handles.editResY,'String'));
dataSelect =
get(handles.popupmenuData,'Value');
axisBasedData = handles.axisBasedData;

if axisBasedData == true
    axisBasedData = 1;
    resolution = resolutionx;
else
    axisBasedData = 2;
    resolution = resolutiony;
end

% DEM 1D
DEM = sortrows(DEM,axisBasedData);
a=1;
zb_init = [];
for x = 2:length(DEM(:,axisBasedData))
    if  DEM(x,axisBasedData)==DEM(x-
        1,axisBasedData)
        b=x;
    else
        if dataSelect == 1
            tempZb = mean(DEM(a:b,3));
        elseif dataSelect == 2
            tempZb = min(DEM(a:b,3));
```

LAMPIRAN A (LANJUTAN)

```
            else
                tempZb = max(DEM(a:b,3));
            end

            zb_init = cat(1,zb_init,tempZb);
            a=x;
        end
    end

    %untuk data terakhir
    if b==length(DEM)
        if dataSelect == 1
            tempZb = mean(DEM(a:b,3));
        elseif dataSelect == 2
            tempZb = min(DEM(a:b,3));
        else
            tempZb = max(DEM(a:b,3));
        end
        zb_init = cat(1,zb_init,tempZb);
    end

    % DEM 2D
    num_sectionx = (max(DEM(:,1))-
    min(DEM(:,1)))/resolutionx;
    num_sectiony = (max(DEM(:,2))-
    min(DEM(:,2)))/resolutiony;

    DEM(:,1) = (DEM(:,1)-
    min(DEM(:,1)))/resolutionx;
    DEM(:,2) = (DEM(:,2)-
    min(DEM(:,2)))/resolutiony;

    zb_2 = zeros([num_sectionx num_sectiony]);
```

LAMPIRAN A (LANJUTAN)

```
for x=1:length(DEM)
    zb_2(DEM(x,1)+1,DEM(x,2)+1)=DEM(x,3);
end

% BATAS:Memulai data 1D pada titik tertinggi
[maks,barismaks] = max(zb_init);
if barismaks >= length(zb_init)-barismaks
    zb_init = zb_init(1:barismaks);
    if axisBasedData == true %sumbu x
        zb_2 = zb_2(1:barismaks,:);
    else %sumbu y
        zb_2 = zb_2(:,1:barismaks);
    end
else
    zb_init =
zb_init(barismaks:length(zb_init));
    if axisBasedData == true %sumbu x
        zb_2 =
zb_2(barismaks:length(zb_init),:);
    else %sumbu y
        zb_2 =
zb_2(:,barismaks:length(zb_init));
    end
end

minX = min(handles.DEM(:,1));
minY = min(handles.DEM(:,2));
[m n] = size(zb_2);
% SAVE 1D
handles.resolution = resolution;
handles.zb_init = zb_init;
handles.min1 =
min(handles.DEM(:,axisBasedData));
```

LAMPIRAN A (LANJUTAN)

```
% SAVE 2D
handles.zb_2 = zb_2;
handles.resolutionx = resolutionx;
handles.resolutiony = resolutiony;
handles.minX = minX;
handles.minY = minY;

%SETTING SLIDER
L = length(zb_init)-1;
set(handles.slider1, 'Min', 0);
set(handles.slider1, 'Max', L);
set(handles.slider1, 'SliderStep', [1 1]);
set(handles.slider1, 'Value', round(L/2));

set(handles.slider2, 'Min', 0);
set(handles.slider2, 'Max', L);
set(handles.slider2, 'SliderStep', [1 1]);

set(handles.slider3, 'Min', 0);
set(handles.slider3, 'Max', L);
set(handles.slider3, 'SliderStep', [1 1]);

if zb_init(1)>zb_init(length(zb_init))
    set(handles.slider2, 'Value', 0);
    set(handles.slider3, 'Value', L);
else
    set(handles.slider2, 'Value', L);
    set(handles.slider3, 'Value', 0);
end
```

LAMPIRAN A (LANJUTAN)

```
%SETTING COORDINATE
if axisBasedData == true
    set(handles.editXMin, 'Enable', 'on');
    set(handles.editXMax, 'Enable', 'on');
    set(handles.editYMin, 'Enable', 'off');
    set(handles.editYMax, 'Enable', 'off');

    set(handles.editYMin,'String',
num2str(0+minY));
    set(handles.editYMax,'String',
num2str((m-1)*resolutiony+minY));
else
    set(handles.editYMin, 'Enable', 'on');
    set(handles.editYMax, 'Enable', 'on');
    set(handles.editXMin, 'Enable', 'off');
    set(handles.editXMax, 'Enable', 'off');

    set(handles.editXMin,'String',
num2str(0+minX));
    set(handles.editXMax,'String',
num2str((n-1)*resolutionx+minX));
end

%PLOT 1D
Plot1D(handles);

%PLOT 2D
X = ((1:m)*resolutionx)+minX;
Y = ((1:n)*resolutiony)+minY;
axes(handles.graf2);
imagesc(X,Y,zb_2');
axis xy;
```

LAMPIRAN A (LANJUTAN)

```
axis tight;
title('Kontur awal lereng (tampak atas)');
xlabel('Posisi X');
ylabel('Posisi Y');

guidata(hObject, handles);
```

4. Fungsi Plot 1D

```
function Plot1D(handles)
% AMBIL DATA
min1 = handles.min1;
zb_init = handles.zb_init;
resolution = handles.resolution;
resolutionx = handles.resolutionx;
resolutiony = handles.resolutiony;
axisBasedData = handles.axisBasedData;
batas1 = round(get(handles.slider1,'Value'));
batas2 = round(get(handles.slider2,'Value'));
batas3 = round(get(handles.slider3,'Value'));

slider1min = (get(handles.slider1,'Min'));
slider1max = (get(handles.slider1,'Max'));
slider2min = (get(handles.slider2,'Min'));
slider2max = (get(handles.slider2,'Max'));
slider3min = (get(handles.slider3,'Min'));
slider3max = (get(handles.slider3,'Max'));

% SETTING COORDINATE
set(handles.editStartPoint,'String',num2str(b
atas1*resolution+min1));
set(handles.editEndPoint,'String',num2str(bat
as2*resolution+min1));
```

LAMPIRAN A (LANJUTAN)

```
if axisBasedData == true
    resolution = resolutionx;

set(handles.editXMin,'String',num2str(batas3*
resolution+min1));

set(handles.editXMax,'String',num2str(batas1*
resolution+min1));
else
    resolution = resolutiony;

set(handles.editYMin,'String',num2str(batas3*
resolution+min1));

set(handles.editYMax,'String',num2str(batas1*
resolution+min1));
end

% SETTING BATAS
set(handles.editBatas1,'String',num2str(batas
1*resolution));
set(handles.editBatas2,'String',num2str(batas
2*resolution));
set(handles.editBatas3,'String',num2str(batas
3*resolution));

% PLOT
axes(handles.graf1);
range = ((0:length(zb_init)-
1)*resolution)+min1;
plot(range,zb_init,'linewidth',2);
axis tight;
title('Keadaan awal lereng (tampak
samping)');
xlabel('jarak dari puncak');
ylabel('Elevasi');
hold all;
```

LAMPIRAN A (LANJUTAN)

```
stem(batas1*resolution+min1,max(zb_init),'--
rd','MarkerEdgeColor','k','MarkerFaceColor','
g','MarkerSize',10);
stem(batas2*resolution+min1,max(zb_init),'--
r<','MarkerEdgeColor','k','MarkerFaceColor','
g','MarkerSize',10);
stem(batas3*resolution+min1,max(zb_init),'--
r>','MarkerEdgeColor','k','MarkerFaceColor','
g','MarkerSize',10);
hold off;
```

5. Fungsi Tampilkan 1

```
function btnTampilkan1_Callback(hObject,
eventdata, handles)
batas1 = round(get(handles.slider1,'Value'));
batas2 = round(get(handles.slider2,'Value'));
resolution = handles.resolution;
min1 = (batas1)*resolution+handles.min1;
zb_init = handles.zb_init(batas1+1 :
batas2+1);
range = ((0:length(zb_init)-
1)*resolution)+min1;

figx = figure;
plot(range,zb_init,'linewidth',2);
set(figx, 'Name', 'Data 1D');
set(figx, 'NumberTitle', 'off');
axis tight;
title('Keadaan awal lereng (tampak
samping)');
xlabel('Jarak dari puncak');
ylabel('Elevasi');
```

LAMPIRAN A (LANJUTAN)

6. Fungsi Simpan 1

```
function btnSimpan1D_Callback(hObject,
eventdata, handles)
[file,path] = uiputfile('*.x1d','Save Data
1D');
if (file == 0)
    return;
end

batas1 = round(get(handles.slider1,'Value'));
batas2 = round(get(handles.slider2,'Value'));
resolution = handles.resolution;
min1 = (batas1)*resolution+handles.min1;
zb_init = handles.zb_init(batas1+1 :
batas2+1);



filename1 = strcat(path,file);
save (filename1, 'min1', 'zb_init',
'resolution');
```

7. Fungsi Tampilkan 2

```
function btnTampilkan2_Callback(hObject,
eventdata, handles)
batas1 = round(get(handles.slider1,'Value'));
batas3 = round(get(handles.slider3,'Value'));

minX = handles.minX;
minY = handles.minY;
resolutionx = handles.resolutionx;
resolutiony = handles.resolutiony;
zb_2 = handles.zb_2;
```

LAMPIRAN A (LANJUTAN)

```
if handles.axisBasedData == true %sumbu x
    zb_2 = handles.zb_2(batas3+1:batas1+1,:);
    zb_2( :, ~any(zb_2,1) ) = [];  %delete
empty columns
    [m,n] = size(zb_2);
    minX = batas3*resolutionx+minX;
    X = ((0:batas1-batas3)*resolutionx)+minX;
    Y = ((1:n)*resolutiony)+minY;
else %sumbu y
    zb_2 = handles.zb_2(:,batas3+1:batas1+1);
    zb_2( ~any(zb_2,2), : ) = [];  %delete
empty rows
    [m,n] = size(zb_2);
    X = ((1:m)*resolutionx)+minX;
    minY = batas3*resolutiony+minY;
    Y = ((0:batas1-batas3)*resolutiony)+minY;
end

figx = figure;
imagesc(X,Y,zb_2');
axis xy;
set(figx, 'Name', 'Data 2D');
set(figx, 'NumberTitle', 'off');
shading flat
title('Kontur awal lereng (tampak atas)');
xlabel('Posisi X');
ylabel('Posisi Y');

figure; surfl(X,Y,zb_2');
```

LAMPIRAN A (LANJUTAN)

8. Fungsi Simpan 2

```
function btnSimpan2D_Callback(hObject,
eventdata, handles)
[file,path] = uiputfile('*.x2d','Save Data
1D');
if (file == 0)
    return;
end
batas1 = round(get(handles.slider1,'Value'));
batas3 = round(get(handles.slider3,'Value'));
minX = handles.minX;
minY = handles.minY;
resolutionx = handles.resolutionx;
resolutiony = handles.resolutiony;
zb_2 = handles.zb_2;


minX = batas3*resolutionx+minX;
minY = batas3*resolutiony+minY;
if handles.axisBasedData == true %sumbu x
    zb_2 = handles.zb_2(batas3+1:batas1+1,:);
    zb_2( :, ~any(zb_2,1) ) = [];  %delete
empty columns
else %sumbu y
    zb_2 = handles.zb_2(:,batas3+1:batas1+1);
    zb_2( ~any(zb_2,2), : ) = [];%delete
empty rows
end
filename1 = strcat(path,file);
save (filename1, 'minX', 'minY',
'resolutionx', 'resolutiony', 'zb_2');
```

LAMPIRAN A (LANJUTAN)

A.2 Kode Program Untuk Pengolahan Data Simulasi

1. Fungsi Open

```
function btnOpenData1_Callback(hObject,
eventdata, handles)
[file,path] = uigetfile('*.x1d','Open Data
1D');
if (file == 0)
    return;
end
filename1 = strcat(path,file);
load (filename1, '-mat');  %  'min1',
'zb_init', 'resolution'

num_section = length(zb_init);
L = (num_section-1)*resolution;
height = zb_init(num_section)-zb_init(1);

if zb_init(1)<zb_init(num_section)
    for x = 1:num_section
        temp(x) = zb_init(num_section+1-x);
    end
    zb_init = temp;
    handles.change_zb_init = true;
else
    handles.change_zb_init = false;
end

%SHOW DATA
set(handles.editL,'String', L);
set(handles.editJmlPias,'String',
num_section);
set(handles.tableDam,'Data', {});
% SAVE DATA
handles.num_section = num_section;
handles.dx_1 = resolution; %%
```

LAMPIRAN A (LANJUTAN)

```
handles.L = L;
handles.min1 = min1 ; %%
handles.zb_init = zb_init; %%
handles.zb_init_ori = zb_init;
handles.er_dpth =
zeros(size(handles.zb_init)); %inisialisasi
er_dpth
handles.open1 = true;
% PLOT 1D
UpdatePlot1D(handles);

guidata(hObject, handles);
```

2. Fungsi Upadate Plot Satu Dimensi

```
function UpdatePlot1D(handles)
% buat plot 1D
zb_init = handles.zb_init;
dx_1 = handles.dx_1;
min1 = handles.min1;

axes(handles.graf1);
range = ((0:length(zb_init)-1)*dx_1)+min1;
if handles.change_zb_init == true
    range = fliplr(range);
end

plot(range,zb_init,'linewidth',2);
axis tight;
title('Keadaan awal lereng (tampak
samping)');
xlabel('jarak dari puncak');
ylabel('Elevasi');
```

LAMPIRAN A (LANJUTAN)

3. Fungsi Tambah Data

```
function btnTambahData_Callback(hObject,
eventdata, handles)
user_response = InputDam;
if (user_response.retOK)
    Ld = user_response.Ld;
    B1 = user_response.B1;
    B2 = user_response.B2;
    H = user_response.H;

    retVal = false;

    if (strcmp(user_response.Type,'Natural'))
        retVal = InsertNaturalDam(hObject,
handles, Ld, B1, B2, H);
    else
        retVal = InsertSaboDam(hObject,
handles, Ld, B1, B2, H);
    end

    if (retVal)
        data = get(handles.tableDam, 'Data');
        [nrow ncol] = size(data);
        nrow = nrow + 1;

        data(nrow, 1:5) = {Ld B1 B2 H
user_response.Type};
        set(handles.tableDam, 'Data', data);
    end
end
```

LAMPIRAN A (LANJUTAN)

4. Fungsi Insert Dam

```
function retVal = InsertSaboDam(hObject,
handles, Ld, B1, B2, H)
% membaca variabel yang diinputkan sebelumnya
L = handles.L;
dx_1 = handles.dx_1;

% memanggil zb_init dan er_dpth
zb_init = handles.zb_init_ori;
er_dpth = handles.er_dpth;

% X TERDEKAT
x_terdekat = round(Ld/dx_1)*dx_1;

if x_terdekat > Ld
    x2 = x_terdekat;
    x1 = x2 - dx_1;
    HLd = (x2-Ld)*(zb_init(x1/dx_1)-
zb_init(x2/dx_1))/(x2-x1)+zb_init(x2/dx_1);
elseif x_terdekat == Ld
    x2 = x_terdekat + dx_1;
    x1 = x_terdekat - dx_1;
    HLd = zb_init(x_terdekat/dx_1);
else  %x_terdekat < Ld
    x1 = x_terdekat;
    x2 = x_terdekat + dx_1;
    HLd = (x2-Ld) * (zb_init(x1/dx_1)-
zb_init(x2/dx_1))/dx_1 + zb_init(x2/dx_1);
end

M1 = sqrt( (Ld-x1)^2 + (HLd-
zb_init(x1/dx_1))^2 );
```

LAMPIRAN A (LANJUTAN)

```
if B1 == M1
    x_atas = x1;
    y_atas = zb_init(x1/dx_1);
elseif B1 < M1
    x_atas = Ld - B1*(Ld-x1)/M1;
    y_atas = HLd + B1*(zb_init(x1/dx_1)-
HLd)/M1;
elseif B1>M1
    for point1 = x1:-dx_1:dx_1
        M1_new = M1 + sqrt( dx_1^2 +
(zb_init(point1/dx_1)-zb_init(point1/dx_1-
1))^2 );
        if B1 == M1_new
            x_atas = point1 - dx_1;
            y_atas = zb_init(point1/dx-1 -
1);
            break;
        elseif B1 < M1_new
            x_atas = point1 - (B1-
M1)*dx_1/(M1_new-M1);
            y_atas =
zb_init(point1/dx_1)+(B1-
M1)*(zb_init(point1/dx_1-1)-
zb_init(point1/dx_1))/(M1_new-M1);
            break;
        end
        M1 = M1_new;
    end
end

M2 = sqrt( (x2-Ld)^2 + (HLd-
zb_init(x2/dx_1))^2 );
```

LAMPIRAN A (LANJUTAN)

```
if B2 == M2
    x_bawah = x2;
    y_bawah = zb_init(x2/dx_1);
elseif B2 < M2
    x_bawah = Ld + B2*(x2-Ld)/M2;
    y_bawah = HLd - B2*(HLd-
zb_init(x2/dx_1))/M2;
elseif B2>M2
    for point2 = x2:dx_1:L
        M2_new = M2 + sqrt( dx_1^2 +
(zb_init(point2/dx_1)-
zb_init(point2/dx_1+1))^2 );
        if B2 == M2_new
            x_bawah = point2 + dx_1;
            y_bawah = zb_init(point2/dx-1 +
1);
            break;
        elseif B2 < M2_new
            x_bawah = point2 +
B2*dx_1/(M2_new-M2);
            y_bawah = zb_init(point2/dx_1+1)
+ (M2-B2)*(zb_init(point2/dx_1)-
zb_init(point2/dx_1 + 1))/(M2_new-M2);
            break;
        end
        M2 = M2_new;
    end
end
miring = sqrt((x_bawah-x_atas)^2 + (y_atas-
y_bawah)^2);
costeta = (y_atas - y_bawah)/miring;
sinteta = (x_bawah - x_atas)/miring;

x_puncak = Ld + H*costeta;
y_puncak = HLd + H*sinteta;
```

LAMPIRAN A (LANJUTAN)

```
x_ataspos = ceil(x_atas/dx_1);
x_puncakpos = round(x_puncak/dx_1);
x_bawahpos = fix(x_bawah/dx_1);

zb_atas = y_atas +
((x_ataspos:x_puncakpos)*dx_1 - x_atas) *
(y_puncak-y_atas)/(x_puncak-x_atas);
zb_bawah = y_bawah + (x_bawah - (x_bawahpos:-
1:x_puncakpos)*dx_1) * (y_puncak-
y_bawah)/(x_bawah-x_puncak);

er_dpth(x_ataspos:x_puncakpos) =
min(er_dpth(x_ataspos:x_puncakpos),max(0,zb_i
nit(x_ataspos:x_puncakpos) - zb_atas));
er_dpth(x_puncakpos:x_bawahpos) =
min(er_dpth(x_puncakpos:x_bawahpos),max(0,zb_
init(x_puncakpos:x_bawahpos) - zb_bawah));

%update zb_init
zb_init = handles.zb_init;
zb_init (x_ataspos:x_puncakpos) =
max(zb_atas,zb_init (x_ataspos:x_puncakpos));
zb_init (x_bawahpos:-1:x_puncakpos) =
max(zb_bawah,zb_init (x_bawahpos:-
1:x_puncakpos));

% simpan
handles.zb_init = zb_init;
handles.er_dpth = er_dpth;

guidata(hObject,handles);
retVal = true;

% update plot
UpdatePlot1D(handles);
```

LAMPIRAN A (LANJUTAN)

5. Fungsi Hapus Data Dam

```
function btnHapusData_Callback(hObject,
eventdata, handles)

BarisDam =
str2num(get(handles.editBaris,'String'));
data = get(handles.tableDam, 'Data');
[nrow ncol] = size(data);

if BarisDam>nrow||BarisDam<1
    msgbox('pilihan baris dam tidak
ada','kesalahan','error')
    return;
end

if nrow ==1
    data = []
else
    if BarisDam == 1
        data = data(2:nrow,:);
    elseif BarisDam == nrow
        data = data(1:nrow-1,:);
    else
        data = cat(1,data(1:BarisDam-
1,:),data(BarisDam+1:nrow,:));
    end
end

set(handles.tableDam,'Data',data);

UpdateLereng(hObject,handles);
```

LAMPIRAN A (LANJUTAN)

6. Fungsi Update Lereng

```
function UpdateLereng(hObject, handles)
% inisialisasi zb_init dan er_dpth
handles.zb_init = handles.zb_init_ori;
handles.er_dpth = zeros(size(handles.zb_init));
guidata(hObject,handles);
UpdatePlot1D(handles);

% Buat input DAM
data = get(handles.tableDam, 'Data');
% [nrow ncol] = size(data);

for i=1:size(data,1)
    Ld = cell2mat(data(i, 1));
    B1 = cell2mat(data(i, 2));
    B2 = cell2mat(data(i, 3));
    H = cell2mat(data(i, 4));
    Type = data(i, 5);
    retVal = false;
    if (strcmp (Type, 'Natural'))
        retVal = InsertNaturalDam(hObject,
handles, Ld, B1, B2, H);
    else
        retVal = InsertSaboDam(hObject, handles,
Ld, B1, B2, H);
    end
    if (~retVal)
        msgbox('Data Input DAM ada yang
salah','Peringatan','warn');
        return;
    end
end
```

LAMPIRAN A (LANJUTAN)

A.3 Kode Program Untuk Dekomposisi Data Dalam Simulasi Aliran Debris

```
% Function Simulasi
tic
load ('data1d.mat');
load ('data2d.mat');

% untuk memudahkan penulisan
dx = dx_2;
dy = dy_2;

g = 980;
sig = 2.65; %sigma (pers 1.3)
rou = 1; %rho (pers 1.3)
fais = 38.5; %psi (pers 1.5)
cst = 0.6; %cst (pers 1.5)
kf = 0.25; %kf (pers 1.6)
kg = 0.0828; %kg (pers 1.6)
e = 0.85; %e (pers 1.6)

h_crit = d/5;
c_crit = 0.05;

dt = 0.01;
int_step = 10; %looping for
num_step = cal_time/dt; %looping for

% initial data set
h1_old = zeros(1,num_section+2);
h1_new = zeros(1,num_section+2);
c1_old = zeros(1,num_section+1);
c1_new = zeros(1,num_section+1);
M1_old = zeros(1,num_section+1);
M1_new = zeros(1,num_section+1);
zb1_old = zeros(1,num_section+1);
zb1_new = zeros(1,num_section+1);
```

LAMPIRAN A (LANJUTAN)

```
% inisialisasi variabel input
h1_old(1) = h_supply;
h1_new(1) = h_supply;
c1_old(1) = c_supply;
c1_new(1) = c_supply;
M1_old(1) = m_supply;
M1_new(1) = m_supply;
zb1_old(2:num_section+1) = zb_init; %zb1_old(fw)
zb1_old(1) = zb1_old(2) + zb_init(1) -
zb_init(2);
zb1_new(2:num_section+1) = zb_init; %zb1_new(fw)
zb1_new(1) = zb1_old(2) + zb_init(1) -
zb_init(2);

% indeks yang digunakan pada debris1d
bk = 1:num_section-1;
ct = 2:num_section;
fw = 3:num_section+1;
ffw = 4:num_section+2;

if min1<= max1
    range = (min1:dx_1:max1);
else %min1>max1
    range = (min1:-dx_1:max1);
end

%ubah zb2 yang bernilai 0
max_zb2 = max(max(zb2));
I = find (zb2==0);
zb2(I) = max(max(zb2))+h_supply;
% zb2(zb2==0) = nan;

% inisialisasi data 2d
M2 = zeros(num_sectionx,num_sectiony - 1);
N2 = zeros(num_sectionx - 1,num_sectiony);
```

LAMPIRAN A (LANJUTAN)

```
h2_old = zeros(num_sectionx - 1,num_sectiony -
1);
h2_new = zeros(num_sectionx - 1,num_sectiony -
1);

%digunakan untuk contourf levelstep
hmaks = max(max(zb2));
hmin = min(min(zb2));

% ALL
zb2_all = zb2;
n_all = n;
Lx_all = Lx;
Ly_all = Ly;
num_sectionx_all = num_sectionx;
num_sectiony_all = num_sectiony;
dx_all = dx;
dy_all = dy;
M2_all = M2;
N2_all = N2;
h2_old_all = h2_old;
h2_new_all = h2_new;


% Batas
sisa_part = jml_part-1;
%batas
if arah == 1 %kiri
    if sisa_part > 0
        batasx_end = round( (jml_part-
sisa_part)/jml_part *(num_sectionx_all-1)); %
berubah
    else
        batasx_end = num_sectionx_all-1;
    end
```

LAMPIRAN A (LANJUTAN)

```
    batasx_start = 1;
    batasy_start = 1;
    batasy_end = num_sectiony-1;
elseif arah == 2 %kanan
    if sisa_part > 0
        batasx_start = round( sisa_part/jml_part
*(num_sectionx_all-1)); % berubah
    else
        batasx_start = 1;
    end

    batasx_end = num_sectionx-1;
    batasy_start = 1;
    batasy_end = num_sectiony-1;
elseif arah == 3 %atas
    if sisa_part > 0
        batasy_start = round( sisa_part/jml_part
*(num_sectiony_all-1)); % berubah
    else
        batasy_start = 1;
    end
    batasx_start = 1;
    batasx_end = num_sectionx-1;
%     batasy_start = round( sisa_part/jml_part
*(num_sectiony-1)); % berubah
    batasy_end = num_sectiony-1;
elseif arah == 4 %bawah
    if sisa_part > 0
        batasy_end = round( (jml_part-
sisa_part)/jml_part *(num_sectiony_all-1));%
berubah
    else
        batasy_end = num_sectiony_all-1;
    end
    batasx_start = 1;
    batasx_end = num_sectionx-1;
    batasy_start = 1;
```

LAMPIRAN A (LANJUTAN)

```
%     batasy_end = round( (jml_part-
sisa_part)/jml_part *(num_sectiony-1));% berubah
end

batasb  = batasx_start:batasx_end;
batask  = batasy_start:batasy_end;
batasb2 = batasx_start:batasx_end+1;
batask2 = batasy_start:batasy_end+1;


zb2 = zb2(batasb,batask);
n = n(batasb,batask);

[num_sectionx num_sectiony] = size(zb2);
Lx = (num_sectionx-1)*dx_2;
Ly = (num_sectiony-1)*dy_2;
num_sectionx = num_sectionx+1;
num_sectiony = num_sectiony+1;

% inisialisasi data 2d
M2 = zeros(num_sectionx,num_sectiony - 1);
N2 = zeros(num_sectionx - 1,num_sectiony);
h2_old = zeros(num_sectionx - 1,num_sectiony -
1);
h2_new = zeros(num_sectionx - 1,num_sectiony -
1);


k = 1;
for i = 1:num_step/int_step
    for j = 1:int_step

        % debris flow 1d
        % calculate m
```

LAMPIRAN A (LANJUTAN)

```
        h_ct = (h1_old(ct) + h1_old(fw))/2;
%persamaan (1.1)
        c_ct = (c1_old(ct) + c1_old(fw))/2;
%persamaan (1.2)
        roum_ct = (sig - rou)*c_ct + rou;
%persamaan (1.3)

        ghdHdx = g*h_ct.*(h1_old(fw) +
zb1_old(fw) - h1_old(ct) - zb1_old(ct))/dx_1;

        theta = atan((zb1_old(ct) -
zb1_old(fw))/dx_1); %persamaan (1.4)

        ty = (roum_ct -
rou)*g.*h_ct.*cos(theta)*tan(fais/180*pi).*(c_ct
/cst).^(1/5); %persamaan (1.5)

        c_for_f = max(c_ct,c_crit);
        h_for_f = max(h_ct,h_crit);
        f = kf*(1 -
c_for_f).^(5/3)./c_for_f.^(2/3);
        f = f + kg*sig/rou*(1 -
e^2)*c_for_f.^(1/3);
        f = 25/4*f.*(h_for_f/d).^(-2);
%persamaan (1.6)

        % kerjakan v (center, forward, back)
        tdknol = find(h_ct >= h_crit);
        v_ct = zeros(1,num_section-1);
        v_ct(tdknol) =
M1_old(ct(tdknol))./h_ct(tdknol);

        tdknol = find((h1_old(fw) +
h1_old(ffw))/2 >= h_crit);
        v_fw = zeros(1,num_section-1);
```

LAMPIRAN A (LANJUTAN)

```
        v_fw(tdknol) =
M1_old(fw(tdknol))./((h1_old(fw(tdknol)) +
h1_old(ffw(tdknol)))/2);
        v_fw = (v_fw + v_ct)/2;

        tdknol = find((h1_old(ct) +
h1_old(bk))/2 >= h_crit);
        v_bk = zeros(1,num_section-1);
        v_bk(tdknol) =
M1_old(bk(tdknol))./((h1_old(ct(tdknol)) +
h1_old(bk(tdknol)))/2);
        v_bk = (v_bk + v_ct)/2;

        xdx = v_fw.*(M1_old(ct) + M1_old(fw)) +
abs(v_fw).*(M1_old(ct) - M1_old(fw));
        xdx = xdx - v_bk.*(M1_old(bk) +
M1_old(ct)) - abs(v_bk).*(M1_old(bk) -
M1_old(ct));
        xdx = xdx/(2*dx_1);

        M1_new(ct) = M1_old(ct);
        tdknol = find(h_ct >= h_crit);
        if ~isempty(tdknol)
            f_term =
rou./roum_ct(tdknol).*f(tdknol)./(2*h_ct(tdknol)
).*abs(M1_old(ct(tdknol))./h_ct(tdknol));
            bunshi = -ghdHdx(tdknol) -
xdx(tdknol) - ty(tdknol)./roum_ct(tdknol) +
(1/dt - f_term).*M1_old(ct(tdknol));
            bunbo = 1/dt + f_term;
            M1_new(ct(tdknol)) = bunshi./bunbo;
        end

        M1_new(ct) = max(M1_new(ct),0);

        % calculate h, zb, c
```

LAMPIRAN A (LANJUTAN)

```
        mxdx = (M1_new(ct) - M1_new(bk))/dx_1;

        cxdx = M1_new(ct).*(c1_old(ct) +
c1_old(fw)) + abs(M1_new(ct)).*(c1_old(ct) -
c1_old(fw));
        cxdx = cxdx - M1_new(bk).*(c1_old(bk) +
c1_old(ct)) - abs(M1_new(bk)).*(c1_old(bk) -
c1_old(ct));
        cxdx = cxdx/(2*dx_1);

        tdknol = find((h1_old(bk) +
h1_old(ct))/2 >= h_crit);
        v_bk = zeros(1,num_section-1);
        v_bk(tdknol) =
M1_new(bk(tdknol))./((h1_old(bk(tdknol)) +
h1_old(ct(tdknol)))/2);

        tdknol = find((h1_old(ct) +
h1_old(fw))/2 >= h_crit);
        v_fw = zeros(1,num_section-1);
        v_fw(tdknol) =
M1_new(ct(tdknol))./((h1_old(ct(tdknol)) +
h1_old(fw(tdknol)))/2);

        v_ct = (v_bk + v_fw)/2;

        theta = atan((zb1_old(ct) -
zb1_old(fw))/dx_1);
        tan_theta_eq = (sig/rou -
1)*c1_old(ct)*tan(fais/180*pi)./((sig/rou -
1)*c1_old(ct) + 1);
        theta_eq = atan(tan_theta_eq);
        erv = v_ct.*tan(theta - theta_eq);
        erv(find(c1_old(ct) <=0 & erv < 0)) = 0;

        zb1_new(ct) = zb1_old(ct) - erv*dt;
```

LAMPIRAN A (LANJUTAN)

```
        tdknol = find(zb1_new(ct) < zb_init(bk)
- er_dpth(bk));
        zb1_new(ct(tdknol)) =
zb_init(bk(tdknol)) - er_dpth(bk(tdknol));

        h1_new(ct) = h1_old(ct) + (-mxdx +
erv)*dt;
        h1_new(ct) = max(h1_new(ct),0);

        c1_new(ct) = zeros(1,num_section-1);
        ch = c1_old(ct).*h1_old(ct) + (-cxdx +
cst*erv)*dt;
        tdknol = find(h1_new(ct) > 0);
        c1_new(ct(tdknol)) =
ch(tdknol)./h1_new(ct(tdknol));
        c1_new(ct) = max(c1_new(ct),0);
        c1_new(ct) = min(c1_new(ct),cst);

        % lowest h, zb, c, m calculation
        zb1_new(1) = zb1_old(2) + zb_init(1) -
zb_init(2);
        zb1_new(num_section+1) =
zb1_old(num_section+1);
        M1_new(num_section+1) =
M1_new(num_section);
        h1_new(num_section+1) =
h1_new(num_section);
        h1_new(num_section+2) =
h1_new(num_section+1);
        c1_new(num_section+1) =
c1_new(num_section);

        % data renewal
        zb1_old(2:num_section+1) =
zb1_new(2:num_section+1);
```

LAMPIRAN A (LANJUTAN)

```
        c1_old(2:num_section+1) =
c1_new(2:num_section+1);
        h1_old(2:num_section+2) =
h1_new(2:num_section+2);
        M1_old(2:num_section+1) =
M1_new(2:num_section+1);

        % jika sudah tidak ada aliran pada
puncak (untuk debris1d)
        if k*dt >= duration_time
            h1_old(1) = 0;
            h1_new(1) = 0;
            c1_old(1) = 0;
            c1_new(1) = 0;
            M1_old(1) = 0;
            M1_new(1) = 0;
        end

        if h1_old(num_section+2) ~= 0

            if
(~isequal(h2_old(:,1),zeros(size(h2_old,1),1))||
~isequal(h2_old(1,:),zeros(1,size(h2_old,2)))
)&& sisa_part > 0


                sisa_part = sisa_part - 1;

                zb2_all (batasb,batask) = zb2;
                n_all (batasb,batask) = n;
                M2_all (batasb2, batask) = M2;
                N2_all (batasb, batask2) = N2;
                h2_new_all (batasb, batask) =
h2_new;
                h2_old_all (batasb, batask) =
h2_old;
```

LAMPIRAN A (LANJUTAN)

```
                if arah == 1 %kiri
                    if sisa_part > 0
                        batasx_end = round(
(jml_part-sisa_part)/jml_part
*(num_sectionx_all-1)); % berubah
                    else
                        batasx_end =
num_sectionx_all-1;
                    end
                elseif arah == 2 %kanan
                    if sisa_part > 0
                        batasx_start = round(
sisa_part/jml_part *(num_sectionx_all-1)); %
berubah
                    else
                        batasx_start = 1;
                    end
                elseif arah == 3 %atas
                    if sisa_part > 0
                        batasy_start = round(
sisa_part/jml_part *(num_sectiony_all-1)); %
berubah
                    else
                        batasy_start = 1;
                    end
                elseif arah == 4 %bawah
                    if sisa_part > 0
                        batasy_end = round(
(jml_part-sisa_part)/jml_part
*(num_sectiony_all-1));% berubah
                    else
                        batasy_end =
num_sectiony_all-1;
                    end
                end
```

LAMPIRAN A (LANJUTAN)

```
                batasb  =
batasx_start:batasx_end;
                batask  =
batasy_start:batasy_end;
                batasb2 =
batasx_start:batasx_end+1;
                batask2 =
batasy_start:batasy_end+1;


                zb2 = zb2_all(batasb,batask);
                n = n_all(batasb,batask);
                [num_sectionx num_sectiony] =
size(zb2);
                Lx = (num_sectionx-1)*dx_2;
                Ly = (num_sectiony-1)*dy_2;
                num_sectionx = num_sectionx+1;
                num_sectiony = num_sectiony+1;
                M2 = M2_all(batasb2,batask);
                N2 = N2_all(batasb,batask2);
                h2_old =
h2_new_all(batasb,batask);
                h2_new =
h2_old_all(batasb,batask);



%=========================================
            end

            % debris flow 2d
            % menghitung h
            h2_new(:,:) = h2_old(:,:) -
2*dt*((M2(2:num_sectionx,:) - M2(1:num_sectionx
- 1,:))/dx + (N2(:,2:num_sectiony) -
N2(:,1:num_sectiony - 1))/dy);
```

LAMPIRAN A (LANJUTAN)

```
            h2_new(:,:) = max(0,h2_new(:,:));



            % menghitung u(i,j)
            u_bb = zeros(num_sectionx +
1,num_sectiony - 1);
            penyebut = h2_new(:,:) +
h2_old(:,:);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_bb(2:num_sectionx,:) =
(M2(2:num_sectionx,:) + M2(1:num_sectionx -
1,:))./penyebut;
            % u diluar daerah diberikan nilai
nol

            % menghitung M(i,j)
            M_bb = zeros(num_sectionx +
1,num_sectiony - 1);
            M_bb(2:num_sectionx,:) =
(M2(2:num_sectionx,:) + M2(1:num_sectionx -
1,:))/2;
            % batas bagian kiri
            M_bb(1,:) = M2(1,:)/2;
            % batas bagian kanan
            M_bb(num_sectionx + 1,:) =
M2(num_sectionx,:)/2;

            % menghitung d/dx(umM)
            ddx_umM = u_bb(2:num_sectionx +
1,:).*M_bb(2:num_sectionx + 1,:) -
u_bb(1:num_sectionx,:).*M_bb(1:num_sectionx,:);
            ddx_umM = ddx_umM/dx;

            % menghitung v(i+1/2,j+1/2)
```

LAMPIRAN A (LANJUTAN)

```
            v_pp =
zeros(num_sectionx,num_sectiony);
            penyebut = h2_old(1:num_sectionx -
2,1:num_sectiony - 2) + h2_old(1:num_sectionx -
2,2:num_sectiony - 1) + h2_old(2:num_sectionx -
1,1:num_sectiony - 2) + h2_old(2:num_sectionx -
1,2:num_sectiony - 1);
            penyebut = penyebut +
h2_new(1:num_sectionx - 2,1:num_sectiony - 2) +
h2_new(1:num_sectionx - 2,2:num_sectiony - 1) +
h2_new(2:num_sectionx - 1,1:num_sectiony - 2) +
h2_new(2:num_sectionx - 1,2:num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_pp(2:num_sectionx -
1,2:num_sectiony - 1) = 4*(N2(1:num_sectionx -
2,2:num_sectiony - 1) + N2(2:num_sectionx -
1,2:num_sectiony - 1))./penyebut;
            % batas bagian bawah
            penyebut = h2_old(1:num_sectionx -
2,1) + h2_old(2:num_sectionx - 1,1) +
h2_new(1:num_sectionx - 2,1) +
h2_new(2:num_sectionx - 1,1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_pp(2:num_sectionx - 1,1) =
2*(N2(1:num_sectionx - 2,1) + N2(2:num_sectionx
- 1,1))./penyebut;
            % batas bagian atas
            penyebut = h2_old(1:num_sectionx -
2,num_sectiony - 1) + h2_old(2:num_sectionx -
1,num_sectiony - 1) + h2_new(1:num_sectionx -
2,num_sectiony - 1) + h2_new(2:num_sectionx -
1,num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
```

LAMPIRAN A (LANJUTAN)

```
            v_pp(2:num_sectionx -
1,num_sectiony) = 2*(N2(1:num_sectionx -
2,num_sectiony) + N2(2:num_sectionx -
1,num_sectiony))./penyebut;
            % batas bagian kiri
            penyebut = h2_old(1,1:num_sectiony -
2) + h2_old(1,2:num_sectiony - 1) +
h2_new(1,1:num_sectiony - 2) +
h2_new(1,2:num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_pp(1,2:num_sectiony - 1) =
4*N2(1,2:num_sectiony - 1)./penyebut;
            % batas bagian kanan
            penyebut = h2_old(num_sectionx -
1,1:num_sectiony - 2) + h2_old(num_sectionx -
1,2:num_sectiony - 1) + h2_new(num_sectionx -
1,1:num_sectiony - 2) + h2_new(num_sectionx -
1,2:num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_pp(num_sectionx,2:num_sectiony -
1) = 4*N2(num_sectionx - 1,2:num_sectiony -
1)./penyebut;
            % batas bagian pojok kiri atas
            penyebut = h2_old(1,num_sectiony -
1) + h2_new(1,num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_pp(1,num_sectiony) =
2*N2(1,num_sectiony)/penyebut;
            % batas bagian pojok kanan atas
            penyebut = h2_old(num_sectionx -
1,num_sectiony - 1) + h2_new(num_sectionx -
1,num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
```

LAMPIRAN A (LANJUTAN)

```
            v_pp(num_sectionx,num_sectiony) =
2*N2(num_sectionx - 1,num_sectiony)/penyebut;
            % batas bagian pojok kiri bawah
            penyebut = h2_old(1,1) +
h2_new(1,1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_pp(1,1) = 2*N2(1,1)/penyebut;
            % batas bagian pojok kanan bawah
            penyebut = h2_old(num_sectionx -
1,1) + h2_new(num_sectionx - 1,1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_pp(num_sectionx,1) =
2*N2(num_sectionx - 1,1)/penyebut;

            % menghitung M(i+1/2,j+1/2)
            M_pp =
zeros(num_sectionx,num_sectiony);
            M_pp(:,2:num_sectiony - 1) =
(M2(:,1:num_sectiony - 2) + M2(:,2:num_sectiony
- 1))/2;
            % batas bagian bawah
            M_pp(:,1) = M2(:,1)/2;
            % batas bagian atas
            M_pp(:,num_sectiony) =
M2(:,num_sectiony - 1)/2;

            % menghitung d/dy(vmM)
            ddy_vmM =
v_pp(:,2:num_sectiony).*M_pp(:,2:num_sectiony) -
v_pp(:,1:num_sectiony -
1).*M_pp(:,1:num_sectiony - 1);
            ddy_vmM = ddy_vmM/dy;

            % menghitung gh.dH/dx
```

LAMPIRAN A (LANJUTAN)

```
            ghdHdx =
zeros(num_sectionx,num_sectiony - 1);
            ghdHdx(2:num_sectionx - 1,:) =
g*(h2_new(1:num_sectionx - 2,:) +
h2_new(2:num_sectionx - 1,:))/2;
            ghdHdx(2:num_sectionx - 1,:) =
ghdHdx(2:num_sectionx -
1,:).*(zb2(2:num_sectionx - 1,:) +
h2_new(2:num_sectionx - 1,:) -
zb2(1:num_sectionx - 2,:) -
h2_new(1:num_sectionx - 2,:))/dx;
            % batas bagian kiri
            ghdHdx(1,:) = g*h2_new(1,:)/2;
            ghdHdx(1,:) = ghdHdx(1,:).*(zb2(1,:)
+ h2_new(1,:))/dx;
            % batas bagian kanan
            ghdHdx(num_sectionx,:) =
g*h2_new(num_sectionx - 1,:)/2;
            ghdHdx(num_sectionx,:) =
ghdHdx(num_sectionx,:).*(- zb2(num_sectionx -
1,:) - h2_new(num_sectionx - 1,:))/dx;

            % menghitung u(i+1/2,j)
            u_pb =
zeros(num_sectionx,num_sectiony - 1);
            penyebut = h2_new(1:num_sectionx -
2,:) + h2_new(2:num_sectionx - 1,:) +
h2_old(1:num_sectionx - 2,:) +
h2_old(2:num_sectionx - 1,:);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pb(2:num_sectionx - 1,:) =
4*M2(2:num_sectionx - 1,:)./penyebut;
            % batas bagian kiri
            penyebut = h2_new(1,:) +
h2_old(1,:);
```

LAMPIRAN A (LANJUTAN)

```
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pb(1,:) = 2*M2(1,:)./penyebut;
            % batas bagian kanan
            penyebut = h2_new(num_sectionx -
1,:) + h2_old(num_sectionx - 1,:);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pb(num_sectionx,:) =
2*M2(num_sectionx,:)./penyebut;

            % menghitung v(i,j+1/2)
            v_bp = zeros(num_sectionx -
1,num_sectiony);
            penyebut = h2_new(:,1:num_sectiony -
2) + h2_new(:,2:num_sectiony - 1) +
h2_old(:,1:num_sectiony - 2) +
h2_old(:,2:num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_bp(:,2:num_sectiony - 1) =
4*N2(:,2:num_sectiony - 1)./penyebut;
            % batas bagian bawah
            penyebut = h2_new(:,1) +
h2_old(:,1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_bp(:,1) = 2*N2(:,1)./penyebut;
            % batas bagian atas
            penyebut = h2_new(:,num_sectiony -
1) + h2_old(:,num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_bp(:,num_sectiony) =
2*N2(:,num_sectiony)./penyebut;

            % menghitung v(i+1/2,j)
```

LAMPIRAN A (LANJUTAN)

```
            v_pb =
zeros(num_sectionx,num_sectiony - 1);
            v_pb(2:num_sectionx - 1,:) =
(v_bp(1:num_sectionx - 2,1:num_sectiony - 1) +
v_bp(2:num_sectionx - 1,1:num_sectiony - 1) +
v_bp(1:num_sectionx - 2,2:num_sectiony) +
v_bp(2:num_sectionx - 1,2:num_sectiony))/4;
            % batas bagian kiri
            v_pb(1,:) = (v_bp(1,1:num_sectiony -
1) + v_bp(1,2:num_sectiony))/2;
            % batas bagian kanan
            v_pb(num_sectionx,:) =
(v_bp(num_sectionx - 1,1:num_sectiony - 1) +
v_bp(num_sectionx - 1,2:num_sectiony))/2;

            % menghitung n(i+1/2,j)
            n_pb =
zeros(num_sectionx,num_sectiony - 1);
            n_pb(2:num_sectionx - 1,:) =
(n(1:num_sectionx - 2,:) + n(2:num_sectionx -
1,:))/2;
            % batas bagian kiri
            n_pb(1,:) = n(1,:)/2;
            % batas bagian kanan
            n_pb(num_sectionx,:) =
n(num_sectionx - 1,:)/2;

            % menghitung K1
            penyebut =
zeros(num_sectionx,num_sectiony - 1);
            penyebut(2:num_sectionx - 1,:) =
2*((h2_new(1:num_sectionx - 2,:) +
h2_new(2:num_sectionx - 1,:))/2).^(4/3);
            % batas penyebut bagian kiri
            penyebut(1,:) =
2*(h2_new(1,:)/2).^(4/3);
            % batas penyebut bagian kanan
```

LAMPIRAN A (LANJUTAN)

```
            penyebut(num_sectionx,:) =
2*(h2_new(num_sectionx - 1,:)/2).^(4/3);
            % melanjutkan menghitung K1
            penyebut(find(penyebut <= h_crit)) =
Inf;
            K1 = g*(n_pb.^2).*sqrt(u_pb.^2 +
v_pb.^2)./penyebut;

            % menghitung u(i+1/2,j+1/2)
            u_pp =
zeros(num_sectionx,num_sectiony);
            penyebut = h2_old(1:num_sectionx -
2,1:num_sectiony - 2) + h2_old(1:num_sectionx -
2,2:num_sectiony - 1) + h2_old(2:num_sectionx -
1,1:num_sectiony - 2) + h2_old(2:num_sectionx -
1,2:num_sectiony - 1);
            penyebut = penyebut +
h2_new(1:num_sectionx - 2,1:num_sectiony - 2) +
h2_new(1:num_sectionx - 2,2:num_sectiony - 1) +
h2_new(2:num_sectionx - 1,1:num_sectiony - 2) +
h2_new(2:num_sectionx - 1,2:num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pp(2:num_sectionx -
1,2:num_sectiony - 1) = 4*(M2(2:num_sectionx -
1,1:num_sectiony - 2) + M2(2:num_sectionx -
1,2:num_sectiony - 1))./penyebut;
            % batas bagian kiri
            penyebut = h2_old(1,1:num_sectiony -
2) + h2_old(1,2:num_sectiony - 1) +
h2_new(1,1:num_sectiony - 2) +
h2_new(1,2:num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pp(1,2:num_sectiony - 1) =
2*(M2(1,1:num_sectiony - 2) +
M2(1,2:num_sectiony - 1))./penyebut;
```

LAMPIRAN A (LANJUTAN)

```
            % batas bagian kanan
            penyebut = h2_old(num_sectionx -
1,1:num_sectiony - 2) + h2_old(num_sectionx -
1,2:num_sectiony - 1) + h2_new(num_sectionx -
1,1:num_sectiony - 2) + h2_new(num_sectionx -
1,2:num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pp(num_sectionx,2:num_sectiony -
1) = 2*(M2(num_sectionx,1:num_sectiony - 2) +
M2(num_sectionx,2:num_sectiony - 1))./penyebut;
            % batas bagian bawah
            penyebut = h2_old(1:num_sectionx -
2,1) + h2_old(2:num_sectionx - 1,1) +
h2_new(1:num_sectionx - 2,1) +
h2_new(2:num_sectionx - 1,1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pp(2:num_sectionx - 1,1) =
4*M2(2:num_sectionx - 1,1)./penyebut;
            % batas bagian atas
            penyebut = h2_old(1:num_sectionx -
2,num_sectiony - 1) + h2_old(2:num_sectionx -
1,num_sectiony - 1) + h2_new(1:num_sectionx -
2,num_sectiony - 1) + h2_new(2:num_sectionx -
1,num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pp(2:num_sectionx -
1,num_sectiony) = 4*M2(2:num_sectionx -
1,num_sectiony - 1)./penyebut;
            % batas bagian pojok kiri atas
            penyebut = h2_old(1,num_sectiony -
1) + h2_new(1,num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
```

LAMPIRAN A (LANJUTAN)

```
            u_pp(1,num_sectiony) =
2*M2(1,num_sectiony - 1)/penyebut;
            % batas bagian pojok kanan atas
            penyebut = h2_old(num_sectionx -
1,num_sectiony - 1) + h2_new(num_sectionx -
1,num_sectiony - 1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pp(num_sectionx,num_sectiony) =
2*M2(num_sectionx,num_sectiony - 1)/penyebut;
            % batas bagian pojok kiri bawah
            penyebut = h2_old(1,1) +
h2_new(1,1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pp(1,1) = 2*M2(1,1)/penyebut;
            % batas bagian pojok kiri atas
            penyebut = h2_old(num_sectionx -
1,1) + h2_new(num_sectionx - 1,1);
            penyebut(find(penyebut <= h_crit)) =
Inf;
            u_pp(num_sectionx,1) =
2*M2(num_sectionx,1)/penyebut;

            % menghitung N(i+1/2,j+1/2)
            N_pp =
zeros(num_sectionx,num_sectiony);
            N_pp(2:num_sectionx - 1,:) =
(N2(1:num_sectionx - 2,:) + N2(2:num_sectionx -
1,:))/2;
            % batas bagian kiri
            N_pp(1,:) = N2(1,:)/2;
            % batas bagian kanan
            N_pp(num_sectionx,:) =
N2(num_sectionx - 1,:)/2;

            % menghitung d/dx(umN)
```

LAMPIRAN A (LANJUTAN)

```
            ddx_umN =
u_pp(2:num_sectionx,:).*N_pp(2:num_sectionx,:) -
u_pp(1:num_sectionx - 1,:).*N_pp(1:num_sectionx
- 1,:);
            ddx_umN = ddx_umN/dx;

            % menghitung v(i,j)
            v_bb = zeros(num_sectionx -
1,num_sectiony + 1);
            penyebut = h2_new + h2_old;
            penyebut(find(penyebut <= h_crit)) =
Inf;
            v_bb(:,2:num_sectiony) =
(N2(:,2:num_sectiony) + N2(:,1:num_sectiony -
1))./penyebut;
            % v diluar daerah diberikan nilai
nol

            % menghitung N(i,j)
            N_bb = zeros(num_sectionx -
1,num_sectiony + 1);
            N_bb(:,2:num_sectiony) =
(N2(:,1:num_sectiony - 1) +
N2(:,2:num_sectiony))/2;
            % batas bagian bawah
            N_bb(:,1) = N2(:,1)/2;
            % batas bagian atas
            N_bb(:,num_sectiony + 1) =
N2(:,num_sectiony)/2;

            % menghitung d/dy(vmN)
            ddy_vmN = v_bb(:,2:num_sectiony +
1).*N_bb(:,2:num_sectiony + 1) -
v_bb(:,1:num_sectiony).*N_bb(:,1:num_sectiony);
            ddy_vmN = ddy_vmN/dy;
```

LAMPIRAN A (LANJUTAN)

```
            % menghitung gh.dH/dy
            ghdHdy = zeros(num_sectionx -
1,num_sectiony);
            ghdHdy(:,2:num_sectiony - 1) =
g*(h2_new(:,1:num_sectiony - 2) +
h2_new(:,2:num_sectiony - 1))/2;
            ghdHdy(:,2:num_sectiony - 1) =
ghdHdy(:,2:num_sectiony -
1).*(zb2(:,2:num_sectiony - 1) +
h2_new(:,2:num_sectiony - 1) -
zb2(:,1:num_sectiony - 2) -
h2_new(:,1:num_sectiony - 2))/dy;
            % batas bagian bawah
            ghdHdy(:,1) = g*h2_new(:,1)/2;
            ghdHdy(:,1) = ghdHdy(:,1).*(zb2(:,1)
+ h2_new(:,1))/dy;
            % batas bagian atas
            ghdHdy(:,num_sectiony) =
g*h2_new(:,num_sectiony - 1)/2;
            ghdHdy(:,num_sectiony) =
ghdHdy(:,num_sectiony).*(- zb2(:,num_sectiony -
1) - h2_new(:,num_sectiony - 1))/dy;

            % menghitung u(i,j+1/2)
            u_bp = zeros(num_sectionx -
1,num_sectiony);
            u_bp(:,2:num_sectiony - 1) =
(u_pb(1:num_sectionx - 1,1:num_sectiony - 2) +
u_pb(1:num_sectionx - 1,2:num_sectiony - 1) +
u_pb(2:num_sectionx,1:num_sectiony - 2) +
u_pb(2:num_sectionx,2:num_sectiony - 1))/4;
            % batas bagian bawah
            u_bp(:,1) = (u_pb(1:num_sectionx -
1,1) + u_pb(2:num_sectionx,1))/2;
            % batas bagian atas
```

LAMPIRAN A (LANJUTAN)

```
            u_bp(:,num_sectiony) =
(u_pb(1:num_sectionx - 1,num_sectiony - 1) +
u_pb(2:num_sectionx,num_sectiony - 1))/2;

            % menghitung n(i,j+1/2)
            n_bp = zeros(num_sectionx -
1,num_sectiony);
            n_bp(:,2:num_sectiony - 1) =
(n(:,1:num_sectiony - 2) + n(:,2:num_sectiony -
1))/2;
            % batas bagian bawah
            n_bp(:,1) = n(:,1)/2;
            % batas bagian atas
            n_bp(:,num_sectiony) =
n(:,num_sectiony - 1)/2;

            % menghitung K2
            penyebut = zeros(num_sectionx -
1,num_sectiony);
            penyebut(:,2:num_sectiony - 1) =
2*((h2_new(:,1:num_sectiony - 2) +
h2_new(:,2:num_sectiony - 1))/2).^(4/3);
            % batas penyebut bagian bawah
            penyebut(:,1) =
2*(h2_new(:,1)/2).^(4/3);
            % batas penyebut bagian atas
            penyebut(:,num_sectiony) =
2*(h2_new(:,num_sectiony - 1)/2).^(4/3);
            % melanjutkan menghitung K2
            penyebut(find(penyebut <= h_crit)) =
Inf;
            K2 = g*(n_bp.^2).*sqrt(u_bp.^2 +
v_bp.^2)./penyebut;

            % menghitung M
            penyebut = 1/(2*dt) + K1;
```

LAMPIRAN A (LANJUTAN)

```
            penyebut(find(penyebut <= h_crit)) =
Inf;
            M2_kiri = M2(1,:);
            M2_kanan = M2(num_sectionx,:);
            M2 = M2/(2*dt) - ddx_umM - ddy_vmM -
ghdHdx - K1.*M2;
            M2 = M2./penyebut;

            if arah == 1 % sumber dari bagian
kiri
                M2(1,jarakpos) =
M1_old(num_section + 1);
            elseif arah == 2 % sumber dari
bagian kanan
                M2(num_sectionx,jarakpos) = -
M1_old(num_section + 1);
            end

            % menghitung N
            penyebut = 1/(2*dt) + K2;
            penyebut(find(penyebut <= h_crit)) =
Inf;
            N2_bawah = N2(:,1);
            N2_atas = N2(:,num_sectiony);
            N2 = N2/(2*dt) - ddx_umN - ddy_vmN -
ghdHdy - K2.*N2;
            N2 = N2./penyebut;

            %kiri, kanan, atas, bawah
            if arah == 3 % sumber dari bagian
atas
                N2(jarakpos,num_sectiony) = -
M1_old(num_section + 1);
            elseif arah == 4 % sumber dari
bagian bawah
```

LAMPIRAN A (LANJUTAN)

```
                N2(jarakpos,1) =
M1_old(num_section + 1);
            end
            % front tip M -> num_sectionx, N ->
num_sectionx - 1
            [hbrs,hklm] = find(h2_new < 0.001);
            M2((hklm - 1)*num_sectionx + hbrs) =
max(0,M2((hklm - 1)*num_sectionx + hbrs));
            M2((hklm - 1)*num_sectionx + hbrs +
1) = min(0,M2((hklm - 1)*num_sectionx + hbrs +
1));
            N2((hklm - 1)*(num_sectionx - 1) +
hbrs) = max(0,N2((hklm - 1)*(num_sectionx - 1) +
hbrs));
            N2(hklm*(num_sectionx - 1) + hbrs) =
min(0,N2(hklm*(num_sectionx - 1) + hbrs));
            % menghapus nilai h yang kecil
            h2_new(find(h2_new < 0.001)) = 0;
            % data renewal
            h2_new(h2_new==inf) = 0;
%menghindari error karena inf
            h2_old = h2_new;
        end

        % menambah variabel waktu (langkah
terakhir)
        k = k + 1;


    end

    %PLOT 1D
    axes(handles.axes1);
    % hold on;
```

LAMPIRAN A (LANJUTAN)

```
    plot(range,zb1_old(2:num_section+1),
'b',range,zb1_old(2:num_section+1)+h1_old(1:num_
section),'r','linewidth',2);
    axis tight;
    title('Keadaan lereng (tampak samping)');
    xlabel('jarak dari puncak');
    ylabel('Elevasi');
    legend('bed (cal)','free surface
(cal)','Location','NorthWest');

    %PLOT 2D
    if h1_old(num_section+2) ~= 0
        axes(handles.axes21);
        zb2_all (batasb,batask) = zb2;
        h2_old_all (batasb,batask) = h2_old;
        imagesc(zb2_all'+h2_old_all');

        colorbar('location','EastOutside');
        axis tight;
        axis xy;
        title('Kontur lereng (tampak atas)');
        xlabel('Posisi X');
        ylabel('Posisi Y');
        caxis([hmin hmaks]);
    end
    set(handles.editSimulationTime,'String',(k-
1)*dt);

set(handles.editComputationalTime,'String',toc);
    pause(0.01);
end

msgbox('Simulasi selesai');

varargout{1} = handles.output;
```

"Halaman ini sengaja dikosongkan"

LAMPIRAN
BIODATA PENULIS

Penulis memiliki nama lengkap Mukhamad Wiwid Setiawan, lahir di Jombang, 12 Juni 1994. Penulis berasal dari Dusun Kagulan Rt. 01 Rw. 01, Desa Janti, Kec. Mojoagung, Kab. Jombang. Penulis telah menempuh pendidikan formal yaitu SD Negeri 1 Gambiran, SMP Negeri 1 Mojoagung, SMA Negeri Mojoagung. Selanjutnya, Penulis melanjutkan studi di jurusan Matematika ITS, dengan mengambil bidang studi ilmu komputer. Selama menjadi mahasiswa, penulis aktif dalam beberapa organisasi antara lain : HIMATIKA ITS dan UKM Teater Tiyang Alit. Penulis juga mengikuti kepanitiaan acara besar di ITS diantaranya : Festamasio 6 Surabaya, OMITS, dan KNM-17. Selama menjadi mahasiswa penulis juga pernah bekerja menjadi tentor mata pelajaran matematika, asisten dosen, dan junior programmer. Dalam penulisan Tugas Akhir ini, penulis tidak lepas dari kekurangan, untuk itu penulis mengharapkan kritik dan saran mengenai tugas akhir ini yang dapat dikirimkan melalui e-mail m.wiwid.s@gmail.com.

"Halaman ini sengaja dikosongkan"