MODUL SMOOTHED PARTICLE HYDRODYNAMICS MODELLING

ADVANCE GEOCALCULATION FOR INFRASTRUCTURE PROBLEM

BAKRIE INFRASTRUCTURE GEO TECH

PROGRAM STUDI TEKNIK SIPIL

FAKULTAS TEKNIK DAN ILMU KOMPUTER

UNIVERSITAS BAKRIE

JAKARTA

2022

KATA PENGANTAR

Puji syukur kita panjatkan kehadirat Allah SWT yang senantiasa melimpahkan segala rahmat, taufik dan hidayah-Nya sehingga tim penyusun dapat menyelesaikan modul ini yang berjudul ***Smoothed Particle Hydrodynamics Modelling Advance Geocalculation For Infrastructure Problem***.

Modul ini disusun untuk mempersiapkan mahasiswa menghadapi dunia profesional dengan mendukung program Merdeka Belajar Kampus Merdeka dan sebagai upaya mencapai IKU No.02 (Mahasiswa mendapatkan pengalaman di luar kampus)

Pembahasan modul ini dimulai dengan menjelaskan tujuan yang akan dicapai. Modul ini terdiri dari beberapa bagian, yaitu tujuan dilakukannya analisis dengan menggunakan *smoothed particle hydrodynamics* (SPH), teori dasar tentang *smoothed particle hydrodynamics* (SPH), dan metode memodelkan dengan menggunakan *smoothed particle hydrodynamics* (SPH).

Penyusun menyadari bahwa di dalam pembuatan modul masih terdapat kekurangan, untuk itu penyusun membuka saran dan kritik yang bersifat membangun. Mudah-mudahan modul ini memberikan manfaat.

Jakarta, 24 November 2022

Penulis

DAFTAR ISI

KATA PENGANTAR ii

DAFTAR ISI iii

DAFTAR TABEL iv

MODUL 03 1

A. Tujuan Praktikum 1

B. Teori Dasar 1

C. Permodelan *Smoothed Particle Hydrodinamics for Soil* 5

1. Install Linux di Windows dengan WSL 5

2. Install PersianSPH 7

3. *Compile Case* Analisis SPH 8

4. Petunjuk langkah untuk membangun model baru 9

LAMPIRAN 20

DAFTAR PUSTAKA 26

DAFTAR TABEL

Tabel 3. 1 Parameter k dan G pada *Top Soil* 12

MODUL 03

Smoothed Particle Hydrodynamics Modelling

A. Tujuan Praktikum

- Untuk melihat permodelan pergerakan partikel tanah dengan *impact load* menggunakan persiansph dengan persamaan Kerr
- Untuk mengetahui bagaimana cara kerja persiansph
- Untuk melihat pergerakan partikel tanah akibat adanya beban pelat

B. Teori Dasar

Persamaan konservasi massa (Eqs. (3) masih berlaku untuk memperkirakan perubahan kepadatan. Namun, bentuk umum dari persamaan momentum diperlukan untuk menentukan gerakan partikel tanah:

$$\frac{d\boldsymbol{v}_i^\alpha}{dt} = \sum_j m_j \left(\frac{\sigma_i^{\alpha\beta}}{\rho_i^2} + \frac{\sigma_j^{\alpha\beta}}{\rho_j^2} \right) \frac{\partial W_{ij}}{\partial x_i^\beta} + \boldsymbol{f}_i^\alpha$$

dimana $\sigma^{\alpha\beta}$ adalah tensor tegangan yang terdiri dari dua bagian: tekanan isotropik, p yang didefinisikan untuk cairan, dan tegangan geser menyimpang $s^{\alpha\beta}$

$$\sigma^{\alpha\beta} = -p\delta^{\alpha\beta} + s^{\alpha\beta}$$

dimana $\delta^{\alpha\beta}$ sedang Kroneckers delta $\delta^{\alpha\beta} = 1$ kalau $\alpha = \beta$ dan $\delta^{\alpha\beta} = 0$ jika $\alpha = \beta$.

Tekanan isotropik untuk partikel tanah dapat dihitung dengan dua cara: persamaan keadaan atau persamaan konstitutif tanah. Kedua cara ini sepenuhnya dijelaskan oleh Bui et al., dan pendekatan kedua, yang didasarkan pada hubungan stres-ketegangan, digunakan dalam penelitian ini sebagai berikut:

$$p = -\frac{\sigma^{\gamma\gamma}}{3} = K\epsilon^{\gamma\gamma}$$

di mana *K* adalah modulus elastisitas *(spring*); $\varepsilon^{\gamma\gamma}$ adalah regangan volumetrik yang merupakan fungsi dari perpindahan tanah yang sebenarnya. Komponen kedua dalam yang merupakan tegangan geser deviatorik, membutuhkan persamaan konstitutif tanah yang tepat. Dalam makalah ini, model plastik elastis-sempurna, sebagai model konstitutif tanah, akan digunakan untuk mensimulasikan perilaku tanah. Model tanah ini akan diterapkan dalam kerangka SPH dengan memanfaatkan model hipo-elastis dan kriteria hasil yang sesuai untuk tanah. Jenis pemodelan tanah ini awalnya disajikan dalam Bui et al. dan dijelaskan secara ekstensif dalam Bui et al.

Hukum Hooke yang digeneralisasi dapat digunakan untuk menghitung tingkat stres dari laju regangan. Ketika mempertimbangkan masalah deformasi yang besar, tingkat stres yang invarian sehubungan dengan rotasi tubuh kaku harus digunakan (tensor tingkat stres objektif). Dengan demikian, tarif Jaumann digunakan. Akhirnya, tingkat stres dapat ditulis dalam bentuk berikut:

$$\dot{\sigma}^{\alpha\beta} - \sigma^{\alpha\gamma}\dot{\omega}^{\beta\gamma} - \sigma^{\gamma\beta}\dot{\omega}^{\alpha\gamma} = 2G\dot{e}^{\alpha\beta} + K\dot{\epsilon}^{\gamma\gamma}\delta^{\alpha\beta}$$

di mana $\dot{\epsilon}^{\gamma\gamma}$ dan $\dot{e}^{\alpha\beta}$ merupakan laju regangan volumetrik dan

deviatorik, masing-masing. Selanjutnya, $\dot{\epsilon}^{\alpha\beta}$ dan $\dot{\omega}^{\alpha\beta}$ adalah tensor laju regangan dan laju rotasi total yang didefinisikan dalam kerangka SPH sebagai:

$$\dot{\epsilon}^{\alpha\beta} = \frac{1}{2}\left(\frac{\partial v^{\alpha}}{\partial x^{\beta}} + \frac{\partial v^{\beta}}{\partial x^{\alpha}}\right) \Rightarrow \dot{\epsilon}_i^{\alpha\beta} = \frac{1}{2}\sum_j \left(\frac{m_j}{\rho_j}(v_j^{\alpha} - v_i^{\alpha})\frac{\partial W_{ij}}{\partial x_i^{\beta}} + \frac{m_j}{\rho_j}(v_j^{\beta} - v_i^{\beta})\frac{\partial W_{ij}}{\partial x_i^{\alpha}}\right)$$

$$\dot{\omega}^{\alpha\beta} = \frac{1}{2}\left(\frac{\partial v^{\alpha}}{\partial x^{\beta}} - \frac{\partial v^{\beta}}{\partial x^{\alpha}}\right) \Rightarrow \dot{\omega}_i^{\alpha\beta} = \frac{1}{2}\sum_j \left(\frac{m_j}{\rho_j}(v_j^{\alpha} - v_i^{\alpha})\frac{\partial W_{ij}}{\partial x_i^{\beta}} - \frac{m_j}{\rho_j}(v_j^{\beta} - v_i^{\beta})\frac{\partial W_{ij}}{\partial x_i^{\alpha}}\right)$$

di mana subskrip i dan j digunakan untuk menggambarkan partikel SPH tanah sebagai lawan dari partikel air. Teori plastisitas aliran digunakan dalam penelitian ini untuk mempertimbangkan perilaku plastik tanah dalam deformasi besar. Teori ini dicirikan oleh asumsi bahwa ada aturan aliran untuk menentukan jumlah deformasi plastis dalam material. Untuk mendefinisikan timbulnya plastisitas, kriteria hasil yang sesuai harus digunakan, dan deformasi plastis hanya terjadi ketika jalur tegangan bergerak melampaui permukaan hasil. Metode Drucker-Prager dipilih sebagai kriteria hasil, Y, dalam penelitian ini, dan permukaan hasil dijelaskan oleh,

$$Y(I_1, J_2) = \sqrt{J_2} + \alpha I_1 = k$$

$$I_1 = \sigma^{xx} + \sigma^{yy} + \sigma^{zz} \quad \text{and} \quad J_2 = \frac{1}{2}s^{\alpha\beta}s^{\alpha\beta}$$

di mana i1 adalah invarian pertama dari tensor stres, dan J2 menunjukkan invarian kedua dari tensor stres deviatorik. α dan k adalah konstanta Drucker-Prager yang dapat diturunkan dari kondisi regangan

bidang sebagai,

$$\alpha = \frac{\tan\phi}{\sqrt{9 + 12\tan^2\phi}} \quad \text{and} \quad k = \frac{3c}{\sqrt{9 + 12\tan^2\phi}}$$

dimana c dan φ adalah kohesi dan sudut gesekan internal tanah, masing-masing. Karena aturan aliran yang tidak terkait dalam teori plastisitas aliran digunakan dalam penelitian ini, diperlukan fungsi potensial plastik, g, yang dapat didefinisikan oleh,

$$g = \sqrt{J_2} + 3I_1 \sin\psi$$

dimana ψ adalah sudut dilatancy, dan sudut dilatancy nol menunjukkan bahwa bahan tersebut secara plastis tidakdapat ditekan. Akhirnya, persamaan konstitutif tanah untuk partikel i dalam bentuk SPH dapat diringkas sebagai,

$$\frac{d\sigma_i^{\alpha\beta}}{dt} = \sigma_i^{\alpha\gamma}\dot{\omega}_i^{\beta\gamma} + \sigma_i^{\gamma\beta}\dot{\omega}_i^{\alpha\gamma} + 2G\dot{e}_i^{\alpha\beta} + K\dot{\epsilon}_i^{\gamma\gamma}\delta^{\alpha\beta} - \dot{\lambda}_i\left[9K\sin\psi\delta^{\alpha\beta} + \frac{G}{\sqrt{J_2}}s_i^{\alpha\beta}\right],$$

$$\dot{\lambda}_i = \frac{3\alpha K\dot{\epsilon}_i^{\gamma\gamma} + (G/\sqrt{J_2})s_i^{\alpha\beta}\dot{\epsilon}_i^{\alpha\beta}}{27\alpha K\sin\psi + G}$$

di mana λ˙ adalah tingkat yang disebut pengganda plastik, λ, yang tergantung pada keadaan stres dan riwayat beban. Persamaan di atas dapat mengekspresikan perilaku tanah sebagai bahan plastik elastis-sempurna. Namun, karena kesalahan numerik selama perhitungan, yang umumnya ditemukan dalam plastisitas komputasi, keadaan tegangan tanah dapat meninggalkan kisaran elastis. Dalam keadaan seperti itu, algoritma pemetaan pengembalian sering digunakan untuk mengembalikan keadaan tegangan secara numerik ke permukaan hasil.

Keakuratan tetapi juga kemampuan model numerik untuk mensimulasikan perilaku mekanis yang realistis dan deformasi tanah tergantung pada definisi kondisi tegangan awal. Dalam modul ini,

pendekatan yang direkomendasikan oleh Bui dan Fukagawa, yang didasarkan pada konsep redaman Linear Viscous, digunakan untuk menghindari fluktuasi tegangan yang besar karena metode pemuatan gravitasi *self-weight*.

C. Permodelan *Smoothed Particle Hydrodinamics for Soil*

1. Install Linux di Windows dengan WSL

Saat ini, PersianSPH hanya tersedia untuk OS Linux. Maka untuk OS Windows perlu diinstall Linux dengan menggunakan perintah WSL.

1. Install WSL command

Anda sekarang dapat menginstal semua yang Anda butuhkan untuk menjalankan Subsistem Windows untuk Linux (WSL) dengan memasukkan perintah ini di administrator PowerShell atau Windows Command Prompt dan kemudian memulai ulang mesin Anda.

```
PowerShell                                                    Copy
wsl --install
```

Perintah ini akan memungkinkan fitur yang diperlukan untuk menjalankan WSL dan menginstal distribusi Ubuntu Linux. (Distribusi *default* ini dapat diubah).

Pertama kali Anda meluncurkan distribusi Linux yang baru diinstal, jendela konsol akan terbuka dan Anda akan diminta untuk menunggu file dihilangkan kompresnya dan disimpan di mesin Anda.

Catatan :

Perintah di atas hanya berfungsi jika WSL tidak diinstal sama sekali, jika Anda menjalankan wsl --install dan melihat teks bantuan WSL, silakan coba jalankan wsl --list --online untuk melihat daftar distro yang tersedia dan jalankan wsl --install -d <DistroName> untuk menginstal distro. Untuk menghapus instalan WSL, lihat Menghapus instalan WSL versi lama atau membatalkan pendaftaran atau menghapus instalan distribusi Linux.

2. Mengubah distribusi Linux *default* yang diinstal

Secara *default*, distribusi Linux yang diinstal adalah Ubuntu. Ini dapat diubah menggunakan bendera -d.

- Untuk mengubah distribusi yang diinstal, masukkan: wsl --install -d <Distribution Name>. Ganti < Distribution Name > dengan nama distribusi yang ingin Anda instal.
- Untuk melihat daftar distribusi Linux yang tersedia yang tersedia untuk diunduh secara online, masukkan: wsl --list --online atau wsl -l -o.
- Untuk menginstal distribusi Linux tambahan setelah instalasi awal, Anda juga dapat menggunakan perintah: wsl --install -d <Distribution Name>.

Tips

Jika Anda ingin menginstal distribusi tambahan dari dalam baris perintah Linux/Bash (bukan dari PowerShell atau Command Prompt), Anda harus menggunakan .exe dalam perintah: wsl.exe --install -d <Distribution Name> atau untuk mencantumkan distribusi yang tersedia: wsl.exe -l -o.

3. Menyiapkan info pengguna Linux Anda

Setelah Anda menginstal WSL, Anda harus membuat akun pengguna dan kata sandi untuk distribusi Linux yang baru Anda instal.

2. Install PersianSPH

Petunjuk instalasi berikut adalah untuk Debian GNU/Linux (tidak stabil) atau Ubuntu 14.04 (atau yang lebih baru). Untuk menginstal kode dengan mudah di folder apa pun di *Personal Computer* Anda, ikuti instruksi di bawah ini:

1. Membuka jendela terminal baru,
2. Membuat direktori baru (misalnya persiansph di home Anda),
3. Pergi ke direktori yang dibuat,
4. mengunduh file skrip instalasi dengan cara mengetikkan wget https://bitbucket.org/persiansph/ph/raw/tip/persiansph_install.sh atau getcopy https://github.com/mghkorzani/persiansph.git dan jalankan bash persiansph_install.sh,
5. Mengikuti petunjuk yang diberikan dengan jawaban yang sesuai sampai instalasi selesai,
6. Menutup semua jendela terminal untuk memuat ulang variabel lingkungan yang dibuat.

```
username@machine:~$ mkdir persiansph
username@machine:~$ cd persiansph
username@machine:~/persiansph$ wget "https://bitbucket.org/persiansph/sph/
   raw/tip/persiansph_install.sh"
.
.
username@machine:~/persiansph$ bash persiansph_install.sh
.
.
Installation is completed.
Close all terminal windows and open a new one to take effect the defined
   environment variables
Please refer to the tutorial to run a simulation.


username@machine:~/persiansph$ exit
```

3. *Compile Case* Analisis SPH

Bagian ini menjelaskan secara singkat cara menjalankan simulasi. Untuk tujuan ini, aliran Poiseuille dalam 2D, sebagai contoh sederhana, dipilih. Contoh ini disediakan dalam lampiran A dan juga termasuk dalam file yang diinstal. Secara umum, kode ini menggunakan perangkat lunak Cmake, yang merupakan paket bebas lintas *platform* dan sumber terbuka, untuk mengelola proses pembuatan hierarki direktori dan pustaka dependen.

Untuk membangun makefile dan melaksanakan simulasi, ikuti prosedur di bawah ini langkah demi langkah:

1. Edit CMakeList.txt di direktori sumber menggunakan notepad agar sesuai dengan file terlampir dalam lampiran B,
2. Buat direktori baru (misalnya poiseuille di home Anda),
3. Pergi ke direktori yang dibuat dengan cara mengetikkan cd direktori,
4. Mengetikkan cmake . kemudian make untuk menghasilkan makefile,
5. Kompilasi semua file,
6. Jalankan file yang sudah dieksekusi untuk mensimulasikan.

```
username@machine:~$ mkdir poiseuille
username@machine:~$ cd poiseuille
username@machine:~/poiseuille$ cmake $SPH
username@machine:~/poiseuille$ make
username@machine:~/poiseuille$ ./Poiseuille
```

4. Petunjuk langkah untuk membangun model baru

Bagian ini menjelaskan cara membuat file model di PersianSPH. Kode ini tidak memiliki unit untuk semua properti material input. Oleh karena itu, semua unit input harus konsisten di seluruh file cpp input. Misalnya, jika Anda mendefinisikan kecepatan dalam m/s, m/s, sebagai satuan kecepatan, harus digunakan di sisa file. Selain itu, output kecepatan akan menjadi dalam m/s. Oleh karena itu, sangat disarankan bahwa sistem unit yang sesuai, seperti SI, digunakan untuk menghindari galat. Ada beberapa sintaks wajib, yang diperlukan untuk model apa pun, sebagai berikut:

1. Mendefinisikan domain SPH: semua partikel dan fungsi yang ditentukan di sisa kode adalah objek domain SPH ini bernama <nama domain>.

   ```
   SPH::Domain <domain name>;
   ```

2. Dimensi masalah: dapat berupa 2D atau 3D tergantung pada masalah.

   ```
   <domain name>.Dimension = <2 or 3 (an integer value)>;
   ```

   2 → 2D, 3 → 3D

3. Jumlah thread CPU: sintaks ini mendefinisikan jumlah thread yang bekerja secara paralel untuk menjalankan simulasi.
   ```
   <domain name>.Nproc = <a non-zero positive integer value>;
   ```

4. Skema integrasi: sintaks ini memilih skema integrasi Verlet (disarankan) atau Leap-Frog yang dimodifikasi untuk menyelesaikan persamaan kontinum dan momentum.
   ```
   <domain name>.Scheme = <0 or 1 (an integer value)>;
   ```

   0 → The modified Verlet scheme

   1 → The Leap-Frog scheme

5. Jenis kernel: fungsi kernel yang tepat untuk suatu masalah harus dipilih menggunakan perintah ini. Untuk informasi lebih lanjut, lihat Gholami Korzani et al. [2017a]

```
<domain name>.KernelType = <an integer value>;
```

0 → Qubic Spline

1 → Quadratic

2 → Quintic

3 → Gaussian with compact support

4 → Quintic Spline

6. Gravitasi: ini adalah variabel opsional untuk mempertimbangkan percepatan Gravitasi dalam suatu model. Oleh karena itu, perintah ini harus dimasukkan dalam simulasi apa pun yang membutuhkan pemodelan Gravity. Gravitasi ke segala arah dapat dipertimbangkan dengan menambahkan sintaks berikut:

```
<domain name>.Gravity = <a vector value>
```

di mana vektor dapat ditulis dalam salah satu dari dua format ini: Vec3 t(x , y , z) atau x ,y , z.

Cara mensimulasikan media tanah plastik elastis-sempurna. Untuk mempertimbangkan plastisitas, kriteria kegagalan Drucker-Prager digunakan dalam dua bentuk: aturan aliran terkait dan tidak terkait (Gholami Korzani et al. [2017b], Bui et al. [2008], Chen dan Mizuno [1990]). Ini sangat mirip dengan pemodelan padat dengan beberapa modifikasi agar sesuai dengan jenis bahan ini. Untuk mendefinisikan bahan padat dalam model, sintaks berikut harus digunakan:

```
<domain name>.Particles[<particle No>]->Material = 3;
<domain name>.Particles[<particle No>]->Cs = <a double value>;
<domain name>.Particles[<particle No>]->G = <a double value>;
<domain name>.Particles[<particle No>]->K = <a double value>;
<domain name>.Particles[<particle No>]->phi = <Radian (a double value)>;
<domain name>.Particles[<particle No>]->c = <a double value>;
<domain name>.Particles[<particle No>]->psi = <Radian (a double value)>;
<domain name>.Particles[<particle No>]->Fail = <2 or 3 (an integer value)>;
```

2 → associated flow rule,

3 → non-associated flow rule

di mana G dan K adalah modulus geser dan spring material, masing-masing. Phi, c dan psi masing-masing menunjukkan sudut gesekan, kohesi, dan sudut dilatensi tanah. Terlihat bahwa aturan aliran terkait tidak memerlukan definisi sudut dilatensi sementara sudut ini diperlukan untuk aturan aliran yang tidak terkait.

SPH menderita ketidakstabilan tarik untuk mensimulasikan tanah dengan kohesi. Oleh karena itu, perlu menggunakan *Artificial stress/pressure (tensile instability)*. Selain itu, *Artificial viscosity* dapat digunakan untuk meningkatkan stabilitas komputasi.

Anlisis SPH membutuhkan informasi data berupa nilai E dan k sesuai yang selanjutnya akan diinput kedalam code Embankment5 yang terdapat pada folder Old_Tests yang termasuk paket hasil unduhan PersianSPH. Untuk memunculkan kasus yang dipergunakan perlu dilakukan compile ulang. Berikut tahapan yang pelu dilakukan :

1. Melakukan perhitungan nilai E dan K pada data tanah seluruh Indonesia :

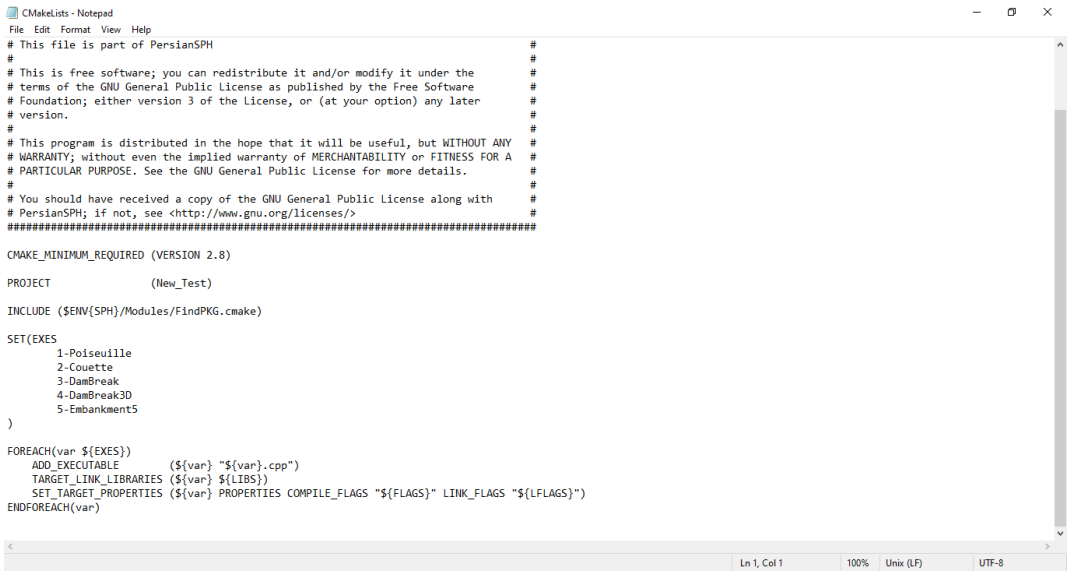
Tabel 3. 1 Parameter k dan G pada *Top Soil*

PARAMETER k DAN G PADA TOP SOIL							
No	Wilayah	Jenis Tanah	G	k1	k2	E1	E2
			kN/m2	N/m2	N/m2	N/m2	N/m2
1	Aceh	Lanau Berkerikil	2964	16260,2	162602	19512,24	195122,40
		Lanau Berpasir	45833,3	223577	2235770	268292,40	2682924,00
2	Jakarta Pusat	Lempung Berkerikil	26000	132144	1321440	158572,80	1585728,00
3	Jawa Barat, Karawang	Lanau	8653,85	45731,7	457317	54878,04	548780,40
		Lanau Berlempung	2962,96	16260,16	162601,6	19512,19	195121,92
4	Jawa Tengah, Semarang	Lanau Berpasir	2962,96	16260,16	162601,6	19512,19	195121,92
5	Jawa Timur, Gresik	Lempung	285,71	1626,02	16260,2	800000,00	8000000,00
6	Kalimantan Selatan, Tarjun	Lempung Berpasir	2962,96	16260,16	162601,6	19512,19	195121,92
		Lempung Berlanau	8653,85	45731,7	457317	22500000,00	225000000,00
7	Wilayah Riau, Dumai	Pasir Berlanau	2173,91	10162,6	101626	12195,12	121951,20
8	Sumatera Utara, Kuala Tanjung	Pasir	6250	30487,8	304878	15000000,00	150000000,00
9	Sumatera Selatan, Palembang	Lempung	285,71	1626,02	16260,2	800000,00	8000000,00
10	Sulawesi Barat, Mamasa	Lempung Berlanau	45833,33	223577,24	2235772,4	268292,69	2682926,88
11	Sulawesi Tengah, Sigi	Lanau Berpasir	2962,96	16260,16	162601,6	19512,19	195121,92
12	Bengkulu	Lempung Berpasir	285,71	1626,02	16260,2	800000,00	8000000,00
		Lempung	8653,85	45731,7	457317	54878,04	548780,40
13	Banten	Lempung	2962,96	16260,16	162601,6	19512,19	195121,92
14	Jawa Tengah, Demak	Lempung	285,71	1626,02	16260,2	800000,00	8000000,00
15	Sumatera Utara, Nassau	Lanau Berpasir	45833,33	223577,24	2235772,4	268292,69	2682926,88
		Lempung Berlanau	2962,96	16260,16	162601,6	19512,19	195121,92

2. Menyalin file Embankment5.cpp dari folder Old_Tests ke folder persiansph-master,

3. Memodifikasi file Embankment5 dengan menggunakan notepad seperti pada lampiran dan mengubah nama file menjadi 5-Embankment5,

4. Edit CMakeList.txt dengan menambahkan contoh kasus yang diperlukan menggunakan notepad seperti pada gambar,



5. Pergi ke direktori yang dibuat dengan cara mengetikkan cd direktori,

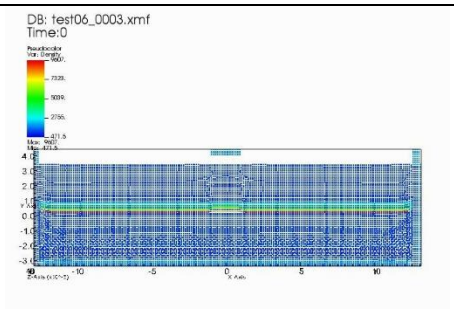
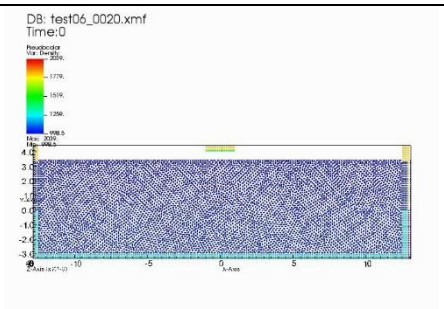
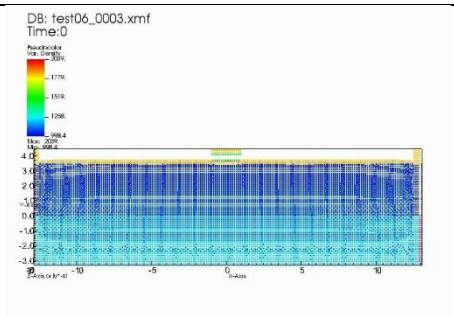
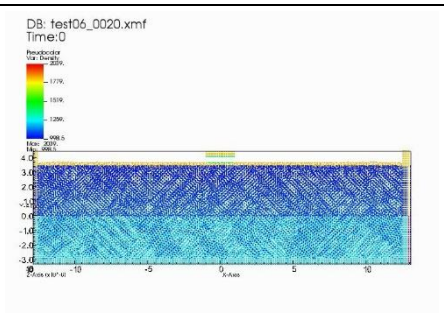
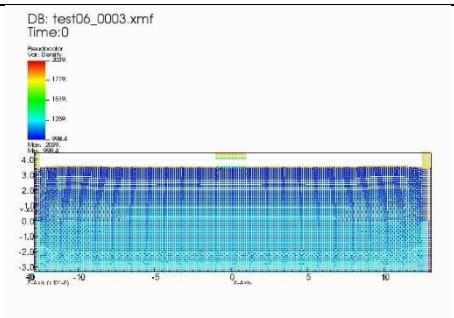
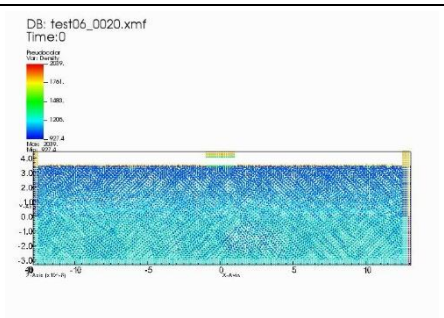
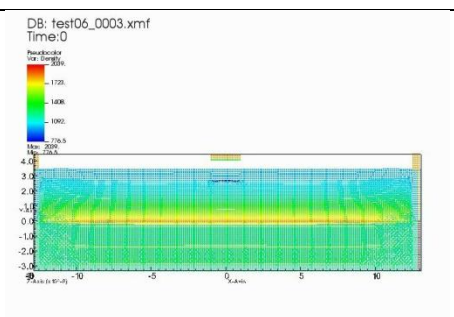
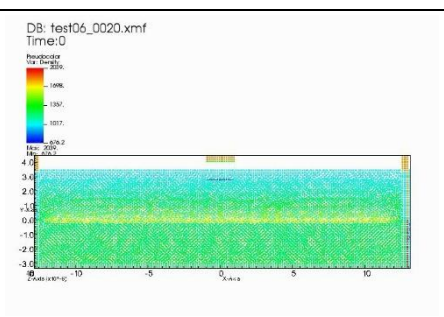
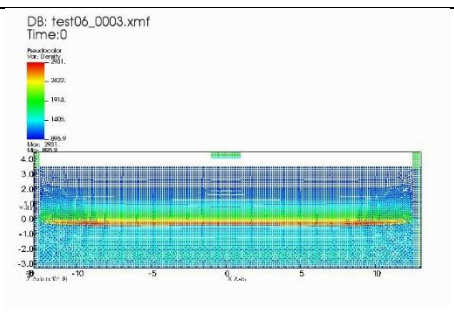
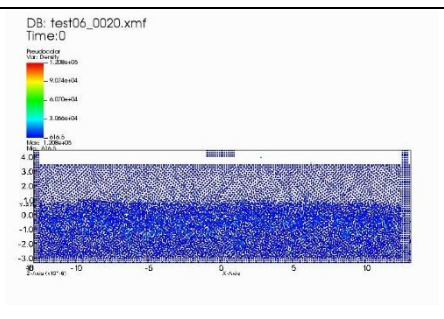
6. Mengetikkan cmake . kemudian make untuk menghasilkan makefile,

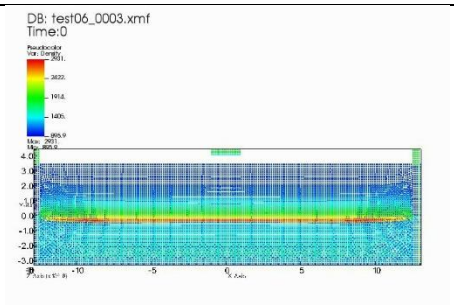
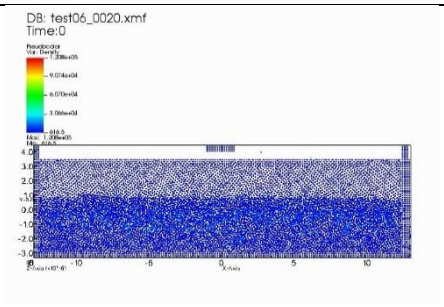
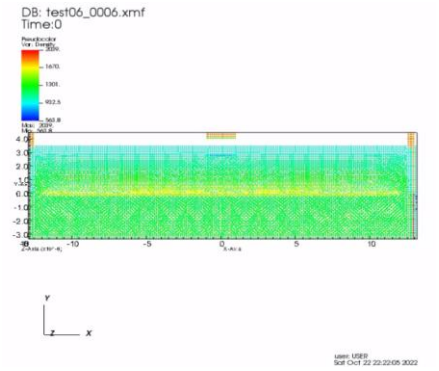
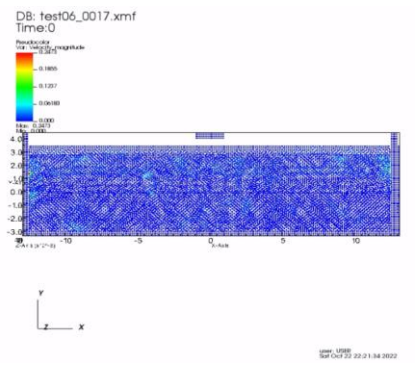
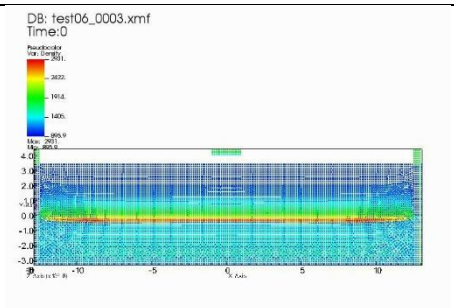
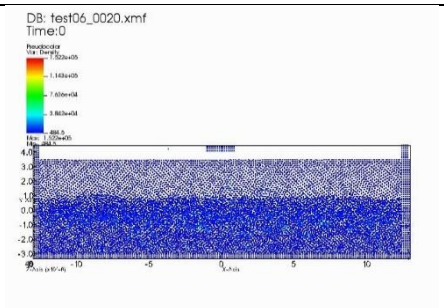
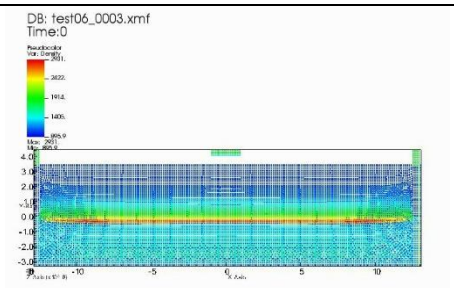
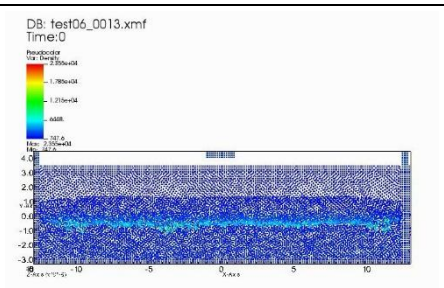
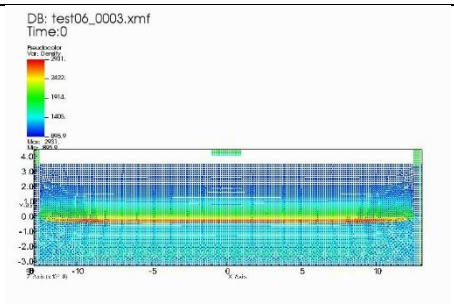
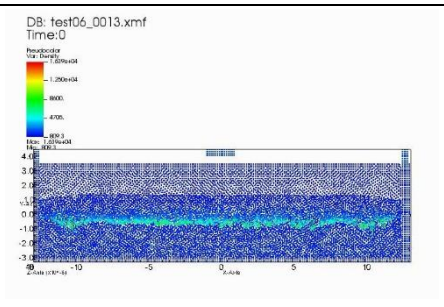
7. Kompilasi semua file,

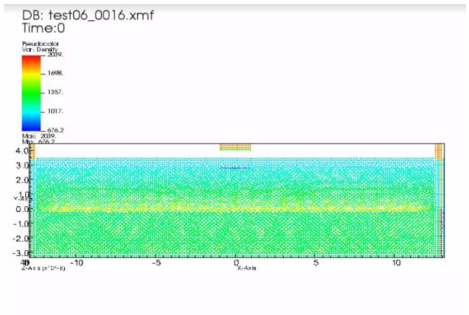
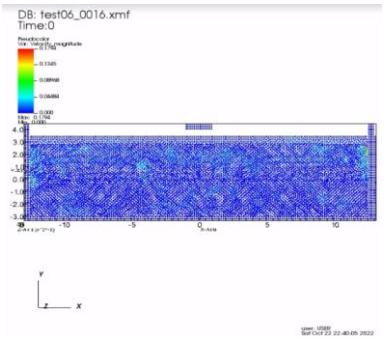
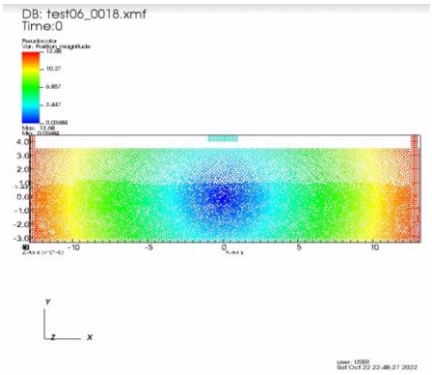
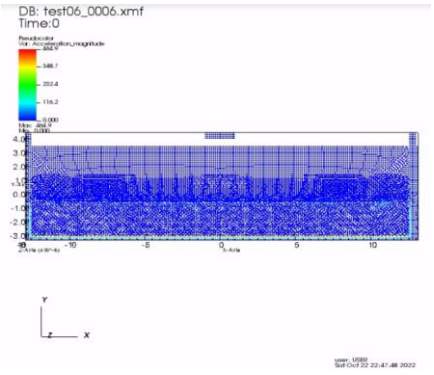
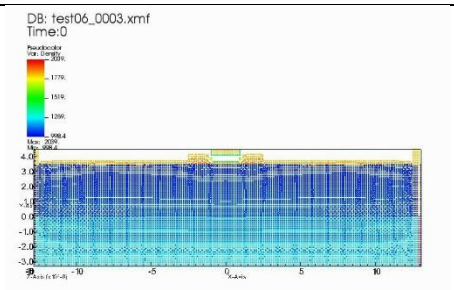
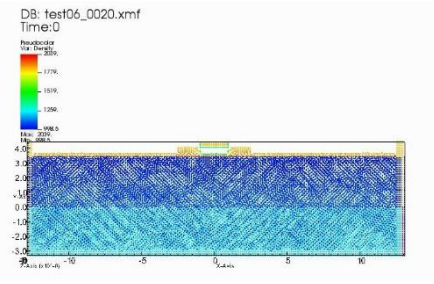
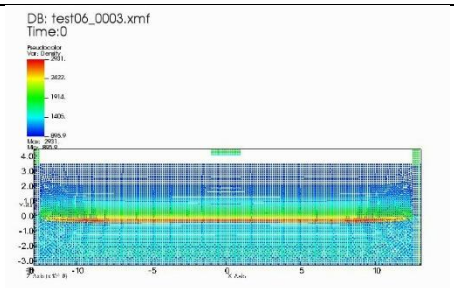
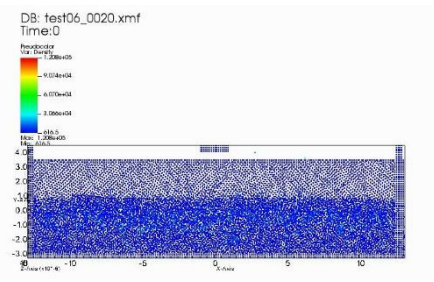
8. Jalankan file yang sudah dieksekusi dengan cara mengetik ./5-Embankment5 untuk mensimulasikan,

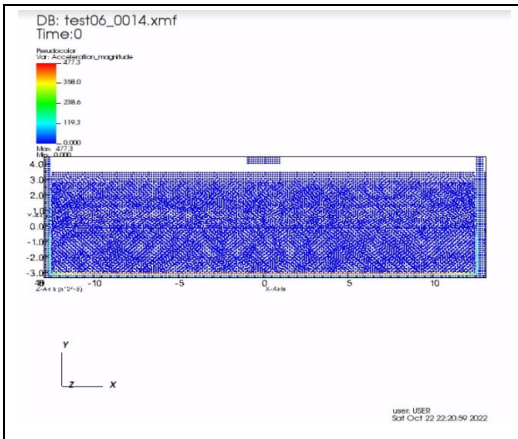
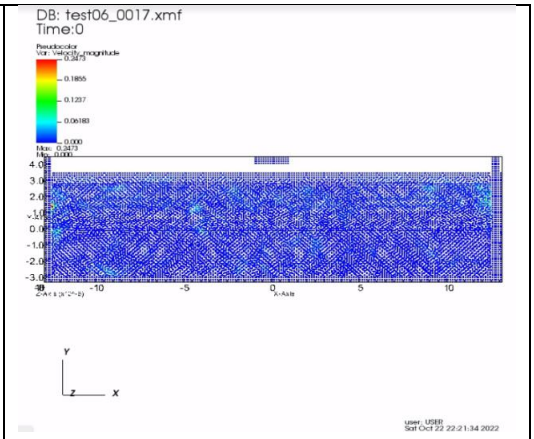
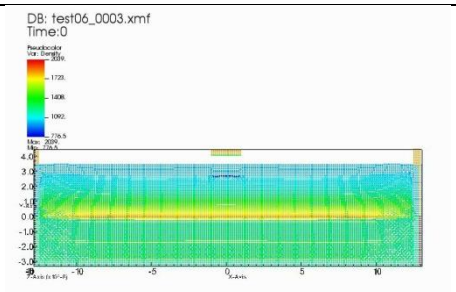
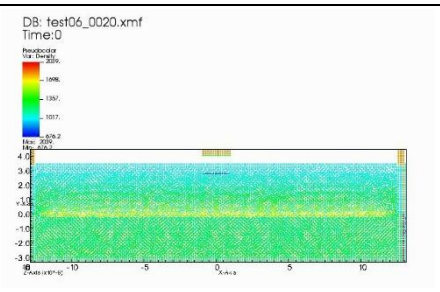
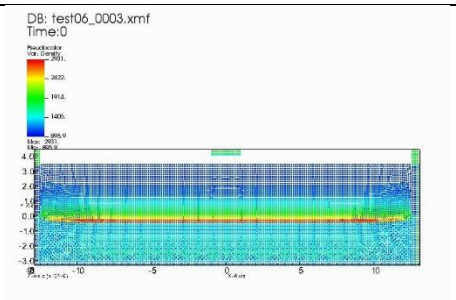
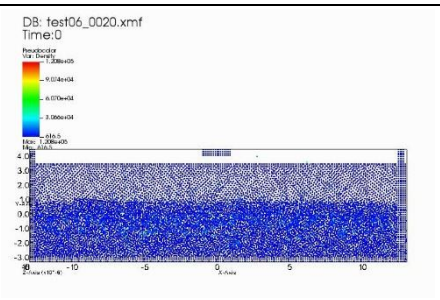
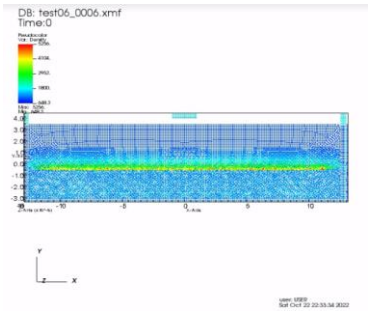
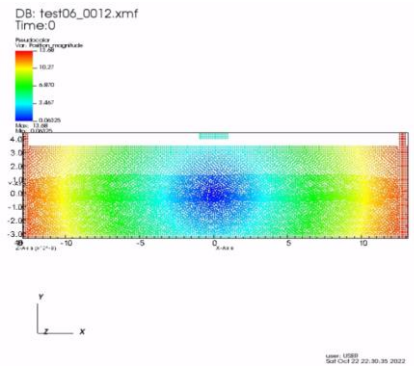
9. Setelah disimulasikan dapat kita visualisasikan menggunakan *software* Paraview maupun Visit.

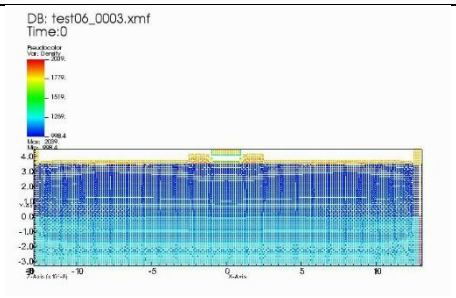
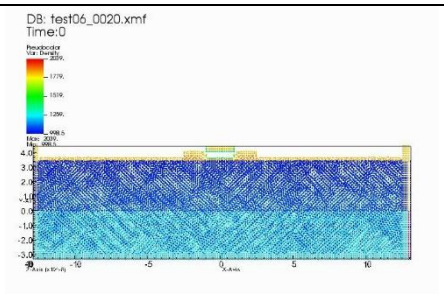
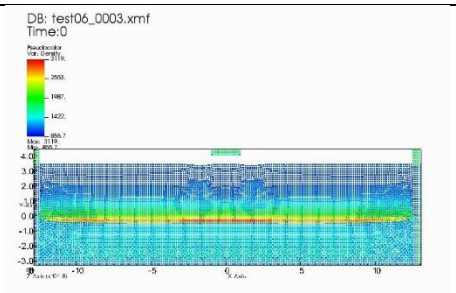
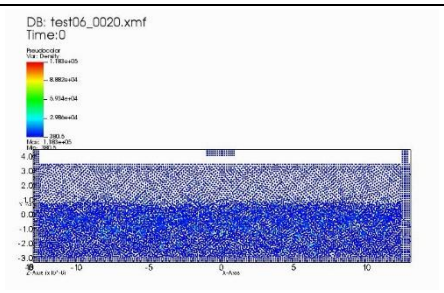
Berikut merupakan hasil simulasi yang telah dilakukan :

Density Tahap 1	*Density* Tahap 2	Jenis Tanah	Daerah
		Lanau Berkerikil	Aceh 01
		Lanau Berpasir	Aceh 02
		Lempung Berkerikil	Jakarta Pusat
		Lanau	Jawa Barat, Karawang 01
		Lanau Berlempung	Jawa Barat, Karawang 02

		Lanau Berpasir	Jawa Tengah, Semarang
		Lempung	Jawa Timur, Gresik
		Lempung Berpasir	Kalimantan Selatan, Tarjun 01
		Lempung Berlanau	Kalimantan Selatan, Tarjun 02
		Pasir Berlanau	Wilayah Riau, Dumai

		Pasir	Sumatera Utara, Kuala Tanjung
		Lempung	Sumatera Selatan, Palembang
		Lempung Berlanau	Sulawesi Barat, Mamasa
		Lanau Berpasir	Sulawesi Tengah, Sigi

		Lempung Berpasir	Bengkulu 01
		Lempung	Bengkulu 02
		Lempung	Banten
		Lempung	Jawa Tengah, Demak

		Lanau Berpasir	Sumatera Utara, Nassau 01
		Lempung Berlanau	Sumatera Utara, Nassau 02

LAMPIRAN

```
/*************************************************************************
*
* PersianSPH - A C++ library to simulate Mechanical Systems (solids, fluids
*
*              and soils) using Smoothed Particle Hydrodynamics method
*
* Copyright (C) 2013 Maziar Gholami Korzani and Sergio Galindo-Torres
*
*
*
* This file is part of PersianSPH
*
*
*
* This is free software; you can redistribute it and/or modify it under the
*
* terms of the GNU General Public License as published by the Free Software
*
* Foundation; either version 3 of the License, or (at your option) any later
*
* version.
*
*
*
* This program is distributed in the hope that it will be useful, but WITHOUT ANY
*
* WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
*
* PARTICULAR PURPOSE. See the GNU General Public License for more details.
*
*
*
* You should have received a copy of the GNU General Public License along with
*
* PersianSPH; if not, see <http://www.gnu.org/licenses/>
*
*************************************************************************
*/

#include "./Source/Domain.h"
#include <fstream>


using std::cout;
using std::endl;

int check = 0;
double DampF,DampS,Cs,u;
double t = 2.0;
double tim = 0.0;
Array< size_t > Forced;
double tout = t,dtout=0.5;
```

```
Vec3_t force,loc;
double tforce = t;

void UserDamping(SPH::Domain & domi)
{
//      if (domi.Time<t)
                #pragma omp parallel for schedule (static) num_threads(domi.Nproc)
                for (size_t i=0; i<domi.Particles.Size(); i++)
                {
                        if (domi.Particles[i]->IsFree &&
domi.Particles[i]->Material == 3) domi.Particles[i]->a -= DampS *
domi.Particles[i]->v;
                        if (domi.Particles[i]->IsFree &&
domi.Particles[i]->Material == 1) domi.Particles[i]->a -= DampF *
domi.Particles[i]->v;

                }

//      if (domi.Time>tforce)
        {
        force = 0.0;
        loc = 0.0;
        Forced.Clear();
                for (size_t i=0; i<domi.Particles.Size(); i++)
                {
                        if (domi.Particles[i]->ID == 7)
                        {
                                Forced.Push(i);
                                force +=
domi.Particles[i]->a*domi.Particles[i]->Mass;
                                loc += domi.Particles[i]->x;
//                              domi.Particles[i]->v    = 0.0,
(domi.Time-tforce)>50.0 ? -0.005 : -0.001 , 0.0;
//                              domi.Particles[i]->vb   = 0.0,
(domi.Time-tforce)>50.0 ? -0.005 : -0.001 , 0.0;
                                domi.Particles[i]->vb   = 0.0 , -0.001 , 0.0;
                                domi.Particles[i]->a    = 0.0 ,  0.0 , 0.0;
                        }
                }
//        if (domi.Time>=tout)
        {
                std::fstream Force ("Force.txt", std::ios::out | std::ios::app );
                Force << force(1) <<std::endl;
                Force.close();
                std::fstream X ("Loc.txt", std::ios::out | std::ios::app );
                X << (loc(1)/Forced.Size()) <<std::endl;
                X.close();
            tout += dtout;
        }
        }
}
```

```
int main(int argc, char **argv) try
{
    SPH::Domain         dom;
        dom.Dimension   = 2;
        dom.Viscosity_Eq_Set(Morris);
        dom.Kernel_Set(Qubic_Spline);
//      dom.KernelType  = 0;
//      dom.SeepageType = 2;
//      dom.VisEq               = 0;
        dom.Nproc               = 8;
//      dom.Excemption  = 4;
//      dom.TimestepConstrain1 = false;
//      dom.XSPH                = 0.5;
        dom.Scheme              = 0;
        dom.Gravity             = 0.0,-9.81,0.0;
        dom.GeneralAfter= & UserDamping;

        double xb,yb,h,dx,T;

        dx              = 0.125;
        h               = dx*1.3;
        dom.InitialDist = dx;

        double rhoF,Mu,CsF,Tf;
        rhoF    = 998.23;
        Mu              = 1.0e-3;
        CsF             = 10.0*5.0;
        Tf              = (0.25*h/CsF);
        Cs              = CsF;
        cout <<"Tf = " << Tf <<endl;
        DampF                   = 0.02*CsF/h;

        dom.AddBoxLength(1 ,Vec3_t (-12.5 - 2.0*dx ,-3.0 - 3.0*dx , 0.0 ), 25.0 +
7.0*dx + dx/10.0 , 6.5 + 3.0*dx + dx/10.0  ,  0 , dx/2.0 ,rhoF, h,1 , 0 ,
false,false);

        for (size_t a=0; a<dom.Particles.Size(); a++)
        {
                dom.Particles[a]->PresEq        = 1;
                dom.Particles[a]->Alpha         = 0.05;
                dom.Particles[a]->Mu            = Mu;
                dom.Particles[a]->MuRef         = Mu;
                dom.Particles[a]->Material      = 1;
                dom.Particles[a]->Cs            = CsF;
//              dom.Particles[a]->Shepard       = true;
//              dom.Particles[a]->ShepardStep = 20;
                xb=dom.Particles[a]->x(0);
                yb=dom.Particles[a]->x(1);
                if (yb<-3.0)
                {
                        dom.Particles[a]->ID    = 2;
                        dom.Particles[a]->IsFree= false;
```

```
                dom.Particles[a]->NoSlip= true;
            }
            if ((xb>12.5 || xb<-12.5))
            {
                dom.Particles[a]->ID    = 2;
                dom.Particles[a]->IsFree= false;
                dom.Particles[a]->NoSlip= false;
            }


            dom.Particles[a]->Density  =
rhoF*pow((1+7.0*9.81*(3.5-dom.Particles[a]->x(1))/(CsF*CsF)),(1.0/7.0));
        }

        double Nu,CsS,Ts,RhoF;
        double E1,K1,G1,CsS1,rhoS1,c1,Phi1,Psi1,d1,n1,k1;
        double E2,K2,G2,CsS2,rhoS2,c2,Phi2,Psi2,d2,n2,k2;

        Nu              = 0.3;
        RhoF    = 8.0e3/9.81;

        E1              = 25.0e6;
        K1              = E1/(3.0*(1.0-2.0*Nu));
        G1              = E1/(2.0*(1.0+Nu));
        rhoS1   = 18.0e3/9.81;
    CsS1        = sqrt(K1/rhoS1);
    c1          = 2.0e3;
    Phi1        = 30.0;
    Psi1        = 0.0;
    d1          = 0.02;
    n1          = 0.35;
        k1              = n1*n1*n1*d1*d1/(150.0*(1-n1)*(1-n1));

        E2              = 50.0e6;
        K2              = E2/(3.0*(1.0-2.0*Nu));
        G2              = E2/(2.0*(1.0+Nu));
        rhoS2   = 20.0e3/9.81;
    CsS2        = sqrt(K2/rhoS2);
    c2          = 10.0e3;
    Phi2        = 20.0;
    Psi2        = 0.0;
    d2          = 0.01;
    n2          = 0.25;
        k2              = n1*n1*n1*d1*d1/(150.0*(1-n1)*(1-n1));

    CsS         = std::max(CsS1,CsS2);
    Ts          = (0.25*h/CsS);
        DampS   = 0.02*sqrt(E1/(rhoS1*h*h));
        cout <<"Ts = " << Ts <<endl;

        T               = std::min(Tf,Ts);
        cout <<"T = " << T <<endl;
```

```
        dom.AddBoxLength(3 ,Vec3_t (-12.5 - 3.0*dx ,-3.0 - 3.0*dx , 0.0 ), 25.0 +
7.0*dx + dx/10.0 , 7.0 + 7.0*dx + dx/10.0  ,  0 , dx/2.0 ,rhoS1, h,1 , 0 ,
false,false);

        for (size_t a=0; a<dom.Particles.Size(); a++)
        {
                if (dom.Particles[a]->ID==3)
                {
//                      dom.Particles[a]->Shepard       = true;
                        dom.Particles[a]->Material      = 3;
                        dom.Particles[a]->Alpha         = 0.1;
                        dom.Particles[a]->Beta          = 0.1;
                        dom.Particles[a]->TI            = 0.5;
                        dom.Particles[a]->TIn           = 2.55;

                        xb=dom.Particles[a]->x(0);
                        yb=dom.Particles[a]->x(1);
                        if (yb<-3.0)
                        {
                                dom.Particles[a]->ID    = 5;
                                dom.Particles[a]->IsFree= false;
                                dom.Particles[a]->NoSlip= true;
                        }
//                      if (yb>(-(1.0/1.5)*(xb-7.5)) && yb>0.0 &&
dom.Particles[a]->ID == 3)
//                              dom.Particles[a]->ID    = 8;
//                      if (yb>( (1.0/1.5)*(xb+7.5)) && yb>0.0 &&
dom.Particles[a]->ID == 3)
//                              dom.Particles[a]->ID    = 8;
                        if ((xb>12.5 || xb<-12.5) && dom.Particles[a]->ID == 3)
                        {
                                dom.Particles[a]->ID            = 6;
                                dom.Particles[a]->IsFree        = false;
                                dom.Particles[a]->NoSlip        = false;
                        }

                        if (yb<=0.0)
                        {
                                dom.Particles[a]->d             = d2;
                                dom.Particles[a]->n             = n2;
                                dom.Particles[a]->k             = k2;
                                dom.Particles[a]->RhoF          = RhoF;
                                dom.Particles[a]->Density       = rhoS2;
                                dom.Particles[a]->Densitya      = rhoS2;
                                dom.Particles[a]->Densityb      = rhoS2;
                                dom.Particles[a]->RefDensity    = rhoS2;
                                dom.Particles[a]->Cs            = CsS2;
                                dom.Particles[a]->G             = G2;
                                dom.Particles[a]->K             = K2;
                                dom.Particles[a]->Fail          = 3;
                                dom.Particles[a]->c             = c2;
                                dom.Particles[a]->phi           = Phi2/180.0*M_PI;
```

```
                                dom.Particles[a]->psi           = Psi2/180.0*M_PI;
                        }
                        else
                        {
                                if (dom.Particles[a]->ID == 3)
                                dom.Particles[a]->ID            = 4;
                                dom.Particles[a]->d             = d1;
                                dom.Particles[a]->n             = n1;
                                dom.Particles[a]->k             = k1;
                                dom.Particles[a]->RhoF          = RhoF;
                                dom.Particles[a]->Cs            = CsS1;
                                dom.Particles[a]->G             = G1;
                                dom.Particles[a]->K             = K1;
                                dom.Particles[a]->Fail          = 3;
                                dom.Particles[a]->c             = c1;
                                dom.Particles[a]->phi           = Phi1/180.0*M_PI;
                                dom.Particles[a]->psi           = Psi1/180.0*M_PI;
                        }
                        if (dom.Particles[a]->ID == 4 && (xb>1.0 || xb<-1.0) &&
yb>=4.0)
                                dom.Particles[a]->ID    = 8;
                }
        }
        dom.DelParticles(8);

        for (size_t a=0; a<dom.Particles.Size(); a++)
        {
                xb=dom.Particles[a]->x(0);
                yb=dom.Particles[a]->x(1);
                if (dom.Particles[a]->ID == 4 && xb<1.0 && xb>-1.0 && yb>=4.0)
                {
                        Forced.Push(a);
                        dom.Particles[a]->ID    = 7;
                }
        }
        std::cout<<Forced<<std::endl;

//      dom.Solve(/*tf*/50000.0,/*dt*/T,/*dtOut*/0.5,"test06",2000);
        dom.Solve(/*tf*/50000.0,/*dt*/T,/*dtOut*/0.5,"test06",20);
        return 0;
}
MECHSYS_CATCH
```

DAFTAR PUSTAKA

Dalrymple, R. A., Gómez-Gesteira, M., Rogers, B. D., Panizzo, A., Zou, S., Crespo, A. J. C., Cuomo, G. and Narayanaswamy, M. (2009). “Smoothed Particle Hydrodynamics for Water Waves, Advances in Numerical Simulation of Nonlinear Water Waves.” World Scientific Publishing.

Gingold, R. A. and Monaghan, J. J. (1977). “Smoothed particle hydrodynamics: Theory and application to non-spherical stars.” *Monthly Notices of the Royal Astronomical Society*, 181, 375-389.

Gray, J. P. and Monaghan, J. J. (2001). “SPH elastic dynamics.” *Comput. Methods Appl. Mech. Engrg.*, 190, 6641-6662.

Korzani, M. G., Galindo-Torres, S. A., Scheuermann, A., & Williams, D. J. (2018). Smoothed Particle Hydrodynamics for investigating hydraulic and mechanical behaviour of an embankment under action of flooding and overburden loads. *Computers and Geotechnics*, *94*, 31-45.

Liu, M. B. and Liu, G. R. (2010). “Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments.” *Archives of Computational Methods in Engineering*, 17, 25-76.

Monaghan, J. J. (1988). “An introduction to SPH.” *Comp. Phys. Comm*., 48, 89-96.

Monaghan, J. J. (1992). “Smoothed particle hydrodynamics.” *Annual Review of Astronomy and Astrophysics*, 30, 543-574.

Rabczuk, T., Eibl, J. and Stempniewski, L. (2003). "Simulation of high velocity concrete fragmentation using SPH/MLSPH." *Int. J. Numer. Methods Eng*., 56, 1421–1444.