Applied Mechanics and Materials Vol 567 (2014) pp 216-221 *Online: 2014-06-06*
© (2014) Trans Tech Publications, Switzerland
doi:10.4028/www.scientific.net/AMM.567.216

Computational Aspects of Submarine Slide Generated Tsunami

Vo Nguyen Phu Huan[1, a], Indra Sati H. Harahap[2, b]

[1]Civil Department, Universiti Teknologi Petronas, Perak, Malaysia

[2]Civil Department, Universiti Teknologi Petronas, Perak, Malaysia

[a]phuhuan129@yahoo.com, [b]indrasati@petronas.com.my

Keywords: tsunami, landslide, parallelization, simulation

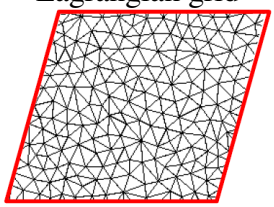
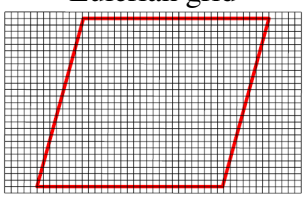
Abstract. Submarine landslide is the most serious threat on both local and regional scales. Tsunami phenomenon induced by submarine slide has put us on the challenge in understanding from generation mechanism to propagation and coastal inundation and mitigating the risk from submarine slide generated tsunami. This research presents the numerical simulation methodology by Smooth Particle Hydrodynamics (SPH) to investigate the impact forces of tsunami waves with the aid of physical modeling. By using parallelSPHysics, it is a source code based on the SPH method to model nearly-incompressible flows, including various physical processes. The conclusions may potentially be taken as guideline of mitigate the risk from tsunami wave.

Introduction

Tsunami can be generated by the energy transfer from submarine slide motion to water motion. The characteristics of a tsunami generated by a submarine landslide are mainly determined by the volume, the initial acceleration, the maximum velocity. Tsunami need to be carefully evaluated before field development can start, as they present threat to human life, environment, seabed installation and drilling operation. However, the tsunami and their impact are not well known.

So far Computational Fluid Dynamics (CFD) has focused on grid-based methods, where two different frames are usually considered for describing the physical governing equations, namely: the Eulerian and the Lagrangian description (Table 1). The grid based numerical methods have achieved remarkably, and they are currently the dominant methods in numerical simulations for solving practical problems in engineering and science.

Table 1. Summary of grid-based method

Lagrangian grid	Eulerian grid
	
✓ Attached on the material ✓ Does not deal with large deformations (remesh) ✓ History of all the variables can be easily obtained ✓ Used by FEM	✓ Fixed in space and with time ✓ Not easy to treat the irregular or complicated geometries ✓ Difficult to tract moving boundary and interface ✓ Used by FDM, FVM

However, the major difficulties are resulted from the use of grid/mesh, which can lead to various difficulties in dealing with problems with free surface, deformable boundary, moving interface, and extremely large deformation and crack propagation. Moreover, for problems with complicated geometry, the generation of a quality mesh has become a difficult, time-consuming and costly process.

Alternative to grid-based methods is a meshfree approach. The development of mesh free methods offers a reliable approach to tackle the simulation of such difficult problems.

Mesh free method

Provide accurate and stable numerical solutions for governing equations with a set of arbitrarily distributed nodes without using any mesh. There are 3 types mesh free method:

- Strong form formulation (simple to implement but instable in some cases).
- Weak form formulation (excellent accuracy but required a background mesh).
- Particle methods.

The state of the system is represented by a set of particles, which possess individual material properties and move according to the governing equations. Smooth Particle Hydrodynamics (SPH) invented by Lucy, Gingold and Monaghan, has been the most popular and successful when applied to coastal hydrodynamic and offshore engineering in general and to free-surface hydrodynamics in particular. It is a mesh free Lagrangian particle method which models fluid flow.

To approximate the values of functions, derivatives at a particle using the information at all the neighboring particles. SPH approximation consists of kernel approximation and particle approximation.

- Kernel approximation: field functions approximated by integral representation method (smoothing effect like weak form)

$$f(x)=\int_{\Omega} f(x_j)W(x-x_j,h)dx_j \tag{1}$$

- Particle approximation: replacing integrations with summations at the neighboring particles in a local domain so-called support domain (sparse matrices).

$$f(x)\approx\sum_{j=1}^{N} m_j \frac{f_j}{\rho_j} W(x-x_j,h) \tag{2}$$

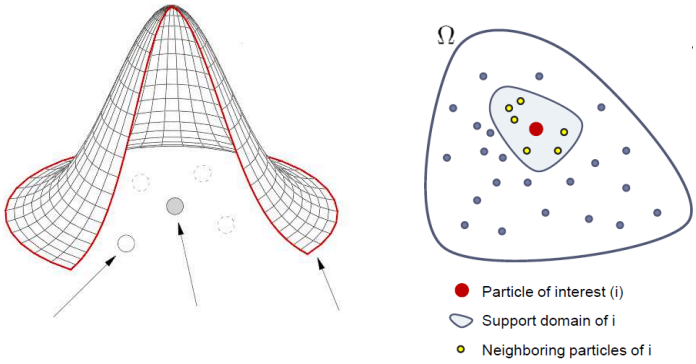


Fig. 1. Particle approximation diagram (Shao, 2009)

By deploying particles at specific positions at the initial stage before the analysis, the free surfaces, material interfaces, and moving boundaries can all be traced naturally in the process of simulation regardless the complicity of the movement of the particles. SPH does not use a grid/mesh. This allows a straight forward handling of very large deformations, since the connectivity between particles are generated as part of the computation and can change with time

Parallelization

❖ Definition of parallelization

Parallel computing is the simultaneous execution of the same task on multiple processors in order to obtain faster results. It is widely accepted that parallel computing is a branch of distributed computing, and puts the emphasis on generating large computing power by employing multiple processing entities simultaneously for a single computation task. These multiple processing entities

can be a multiprocessor system, which consists of multiple processors in a single machine connected by bus or switch networks, or a multicomputer system, which consists of several independent computers interconnected by telecommunication networks or computer networks.

❖ Motivation of parallel computing

The main purpose of doing parallel computing is to solve problems faster or to solve larger problems.

Parallel computing is widely used to reduce the computation time for complex tasks. Many industrial and scientific research and practice involve complex large-scale computation, which without parallel computers would take years and even tens of years to compute. It is more than desirable to have the results available as soon as possible, and for many applications, late results often imply useless results.

While a serial computer can work on one unit area, a parallel computer with N processors can work on N units of area, or to achieve N times of resolution on the same unit area. In numeric simulation, larger resolution will help reduce errors, which are inevitable in floating point calculation; larger problem domain often means more analogy with realistic experiment and better simulation result.

❖ Theoretical model of parallel computing

The widely-used theoretic model of parallel computers is Parallel Random Access Machine (PRAM). A simple PRAM capable of doing add and subtract operation is described in Fortune's paper. A PRAM is an extension to traditional Random Access Machine (RAM) model used to serial computation. It includes a set of processors, each with its own PC counter and a local memory and can perform computation independently. All processors communicate via a shared global memory and processor activation mechanism similar to UNIX process forking. Initially only one processor is active, which will activate other processors; and these new processors will further activate more processors. The execution finishes when the root processor executes a HALT instruction. Readers are advised to read the original paper for a detailed description.

❖ Architectural models of parallel computer

Despite a single standard theoretical model, there exist a number of architectures for parallel computer. Diversity of models is partially shown in Fig 2. This subsection will briefly cover the classification of parallel computers based on their hardware architectures.

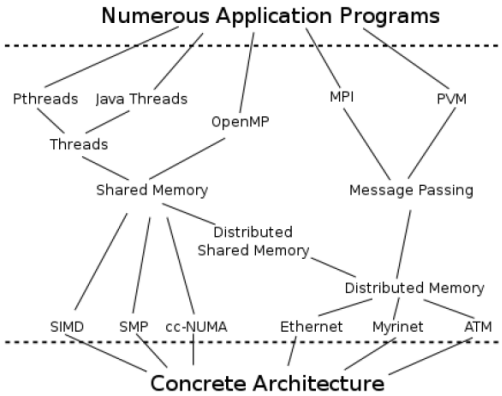


Figure. 2. Diversity of models is partially (Mahantia, 2004)

❖ Hardware and software environments in parallel computing

In this part, we will focus on hardware and software platforms we use in our research.

✓ **Hardware:**

- With the hardware, we have 4 types: Computer Cluster, Computational Grid, Network of Workstations, Vector processing in commodity CPU and GPU. And we propose to use Computational Grid to integrate our system into other components of whole system.
- Computational Gird is designed to integrate distributed computing power to form a single supercomputer, which would be a rather powerful hypercomputer to tackle large-scale computation problems. When message passing programming is to be considered, the

mainstream Message Passing Interface (MPI) implementation, MPICH, has support for Grid-based communication. Unfortunately, in practice there are a lot of problems to be circumvented. For example, for many computer clusters, only the head node has the public Internet Protocol (IP) address, which makes the direct Grid-based communication between 2 client nodes residing in different computer clusters impossible. When our research is to be considered, the heterogeneous nature and hierarchical structure of Gird are further barriers to efficient deployment of our flat-structured MPI program.

✓ **Software:**

- Parallel programming is a complex task. In order to reduce this complexity, different programming models are abstracted, with each providing tools such as special-purpose compilers, libraries and frameworks to simplify programming task. These tools hide many details about parallel execution, such as message transfer and routing, task allocation and migration, and platform differences. Our research is based on message passing programming model and specifically on Message Passing Interface (MPI) standards and MPICH library.
- Message Passing Interface, or MPI, is the most widely-used message passing standard. The basic functions are defined by the MPI standard, and with many implementations targeting almost all distributed memory architectures, it is the industrial standard for message passing programming.
- Basically, MPI provides two types of communication operations. Point-to-point operations allow any two processes to exchange information via MPI_Send (for sending), MPI_Recv (for receiving) and their variants. Collective operations are provided so that a set of processes, known as a communicator, can share and dispatch data through broadcast and reduction operations.
- When an MPI program runs, the user will explicitly specify the number of parallel processes and how the processes are mapped to physical processors. On startup, each processor starts one or more processes to execute the same program body. Each parallel process will be assigned a rank, which serves as the identity of the process, and which will also cause processes to carry out different computation despite their common program body. During the execution, processes carry their own computation, without synchronization with other processes unless they encounter an explicit synchronization command. Processes communicate with each other using point-to-point or collective communication primitives, using process rank to address the recipient or sender if it is required. The whole parallel program exits when all the parallel processes have finished.

Parallel SPHysics

This part shows how to apply parallel computing in SPHYSICS code.

Launching the parallelSPHysics code will depend heavily on the architecture using. The code is parallelized using MPI formalism and thus require MPICH or OpenMPI to be installed on your parallel machine. The code has been developed with the assumption that most parallel environments are using on Linux not on Windows.

❖ SPHYSICSgen: creates the initial conditions and files for a given case.

✓ **Creating compiling options**: The compilation of SPHysics code depends on nature of the problem under consideration and on the particular features of the run. Thus, the user can choose the options that are better suited to any particular problem and only those options will be included in the executable versions of SPHysics. There are only two new options to choose the MPI FORTRAN compiler and whether to activate optimization of the MPI partitioning (i.e. load balancing).

- ✓ **Compiling and Running the executable:** The SPHysics code has been developed on machine that use the LSF (Load Sharing Facility) and the SGE Submission systems. With the code the script file: script_bsub.bsub and script_qsub.qsub have been provide for the LSF and SGE Submission systems. The parallel SPHysics code has been designed to perform simulations of million of particles by using Load Sharing Facility (LSE) and SGE Submission systems.

❖ SPHYSICS: Run the selected case with the initial condition created by SPHYSICgen code.

- ✓ A number of subroutines are necessary to deal with transfer of particles and their information between adjacent processors. The MPI topology must be accessible by each subroutine (i.e. rank, east & west neighbors, MPI communicator, etc.).
- ✓ Grid sweep:
  - Around each cell, the E, N, NW & NE neighboring boxes are checked to minimize repeating the particle interactions. This process is shown schematically in Fig 3.

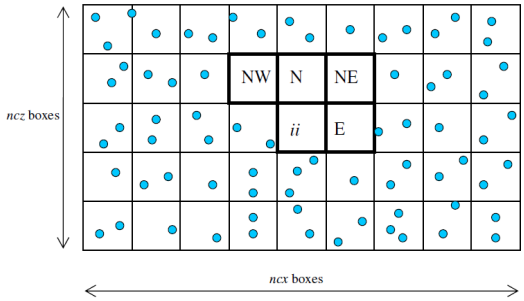


Figure. 3. Grid sweep (SPHysics guide, 2011)

  - This provides an efficient method of identifying the connectivity of the particles with each other. To parallelize the code, the workload needs to be distributed amongst the available processors. Fig 4 displays a typical situation where the domain has been split amongst three different processors.

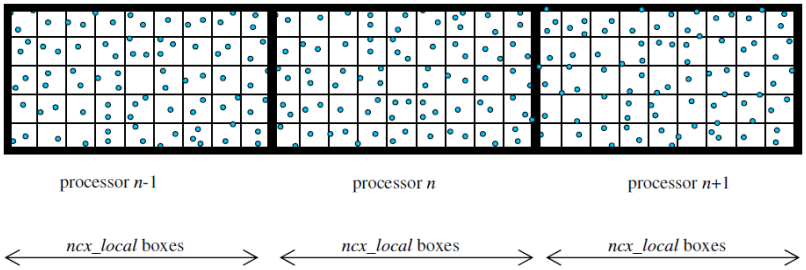


Figure. 4. Domain decomposition in parallel SPHysics (SPHysics guide, 2011)

  - However, when the domain is split up amongst several processors and box ii is located on the boundary of the processor, the code needs to know the contents of box-position ii+1, i.e. E & NE which lie on a different processor. The easiest method to accomplish the necessary transfer of information is to use a column of ghost cells of width 2h as shown in Fig 5.

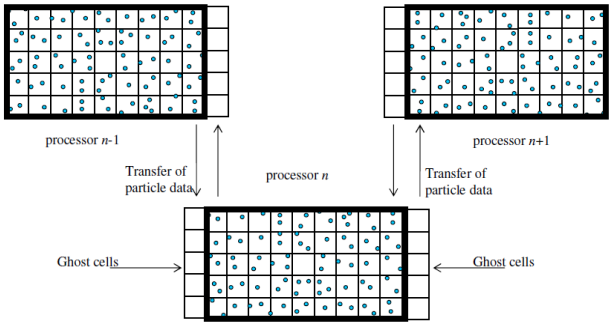


Figure. 5. Importing particle information into ghost cells from neighboring processor (SPHysics guide, 2011)

❖ Some results from parallelSPHysics

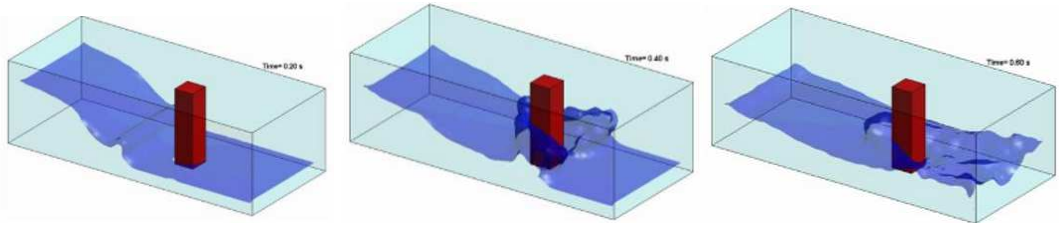


Figure. 6: Tsunami wave hitting a structure (SPHysics test, 2012)

Conclusion

Now, physical understanding of slide tsunami hazards is poor. We must understand substance of tsunami clearly and how to find methods to reduce damage from tsunami wave. This is not simple task because of the complexity and multi scale of process. For this reason, numerical simulation about tsunami wave from submarine slide are very necessary.

To simulating tsunami wave is not simple, because we have some problem with free surface, deformable boundary, moving interface, and extremely large deformation and crack propagation. Moreover, for the problems with complicated geometry, the generation of a quality mesh has become a difficult, time-consuming and costly process.

Alternative to grid-based methods is a mesh free approach. The development of mesh free methods offers a reliable approach to tackle the simulation of such difficult problems. Of the mesh free techniques developed over past decade, Smooth Particle Hydrodynamics (SPH) invented by Lucy, Gingold and Monaghan, has been the most popular and successful when applied to coastal hydrodynamic and offshore engineering in general and to free-surface hydrodynamics in particular. That approach still remains limitations need to be improved related to accuracy and stability, inconsistency problem, or non-conservation of linear and angular momentum. Therefore, the need for comprehensive model base on mesh free method that could cover all aspects of tsunami phenomena and provide real-time modeling of the event would be one of good future research direction.

The conclusions may potentially be taken as guideline of mitigate the risk from tsunami wave.

References

[1] B. Ataie-Ashtiani, G.Shobeyri, “Numerical simulation of landslide impulsive waves by incompressible smoothed particle hydrodynamics”, Int. J. Numer. Meth. Fluids 56:209–232, 2008.

[2] S. Shao, D. I. Graham, C. Ji, D. E. Reeve, P. W. James, and A. J. Chadwick, “Simulation of wave overtopping by an incompressible SPH model”, Coastal Engineering, Vol. 53, No. 9, pp. 723-735, 2009.

[3] J.M. Cherfils, G. Pinona, E. Rivoalena, “A parallel SPH code for free-surface flows”, Computer Physics Communications 183 1468–1480, 2012.

[4] A. Mahantia, D.L. Eagerb, “Adaptive data parallel computing on workstation clusters”, Journal of parallel and distributed computing 64, 1241 – 1255, 2004.

[5] A. Migdalas, G. Toraldo, V. Kumar, “Nonlinear optimization and parallel computing”, Parallel Computing 29,375–391, 2003.

[6] N. Tremblay, M. Florian, “Temporal shortest paths: parallel computing implementations”, Parallel computing 27, 1569 – 1609, 2001.

[7] M. Cannataro, D. Talia, P.K. Srimani, “Parallel data intensive computing in scientific and commercial applications”, parallel Computing 28, 673–704, 2002.

Structural, Environmental, Coastal and Offshore Engineering
10.4028/www.scientific.net/AMM.567

Computational Aspects of Submarine Slide Generated Tsunami
10.4028/www.scientific.net/AMM.567.216