20th International Congress on Modelling and Simulation, Adelaide, Australia, 1–6 December 2013
www.mssanz.org.au/modsim2013

# High Resolution Tsunami Inundation Simulations

**S.G. Roberts** [a], **Y. Oishi** [b], **M. Li** [b]

[a]*Mathematical Sciences Institute, Australian National University, Canberra, ACT 0200*
[b]*Fujitsu Laboratories of Europe, United Kingdom*
*Email: Stephen.Roberts@anu.edu.au*

**Abstract:** In this paper we investigate the high performance computing efficiency of the shallow water software package ANUGA . This package is developed as a collaborative project between the Australian National University (ANU) and Geoscience Australia (GA) and is available as Free and Open Source Software (FOSS). ANUGA uses a shallow water model and approximates the model using the finite volume method based on unstructured meshes of triangles. The geometrical flexibility of unstructured meshes is convenient for tsunami inundation modeling where the tsunami wave source generally consists of long wavelength components, and waves around the coast consists of short wavelengths, which can both be modeled in the same simulation. ANUGA is written in the high level computer language PYTHON. We will present an overview of the model and the numerical method in the early sections of the paper. We will then present our work on parallelizing the ANUGA code, in particular our efforts to obtain efficient simulations using 100s of CPU cores. Our results demonstrate that our PYTHON based software can obtain high efficiency on highly parallel computers. The results presented in this paper demonstrate better than real time simulation of medium resolution (millions of triangles) tsunami models. Our ultimate goal is the solution of high resolution (tens of millions of triangles) simulations in better than real time.

***Keywords:*** *Shallow water wave equations, tsunami simulation, finite volume method, high performance com-puting, python*

## 1 INTRODUCTION

Modelling the effects of natural hazards such as riverine flooding, storm surges and tsunami is critical for an understanding of their economic and social impact on our urban communities. The Australian National University and Geoscience Australia have developed a hydrodynamic inundation modelling tool called ANUGA to help simulate the impact of these hazards.

In the next sections we will review some of our efforts to migrate this tool to modern multicore multinode computer systems.

The core of ANUGA is the fluid dynamics module, which is based on a finite-volume method for solving the Shallow Water Wave Equation. The study area is represented by a mesh of triangular cells. By solving the governing equation within each cell, water depth and horizontal momentum are tracked over time.

A major capability of ANUGA is that it can model the process of wetting and drying as water enters and leaves an area. This means that it is suitable for simulating water flow onto a beach or dry land and around structures such as buildings. ANUGA is also capable of modelling hydraulic jumps due to the ability of the finite-volume method to accommodate discontinuities in the solution.

To set up a particular scenario the user specifies the geometry (bathymetry and topography), the initial water level (stage), boundary conditions such as tide, and any forcing terms that may drive the system such as rainfall, abstraction of water, wind stress or atmospheric pressure gradients. Gravity and frictional resistance from the different terrains in the model are represented by predefined forcing terms.

Most ANUGA components are written in the object-oriented programming language PYTHON. Software written in PYTHON can be produced quickly and can be readily adapted to changing requirements throughout its lifetime. Computationally intensive components are written for efficiency in C routines working directly with Python numeric structures. This combination of interpreted language and compiled code does lead to some challenges (computational efficiency, large memory overhead) when developing efficient code, but does provide an environment which allows users to add new facilities to the code in a simple manner.

See Zoppou and Roberts (1999); Nielsen et al. (2005) for more background on ANUGA . Similar implementations are reported by Chippada et al. (1998), and Begnudelli and Sanders (2007).

## 2 MATHEMATICAL MODEL

The shallow water wave equations are a system of differential conservation equations which describe the flow of a thin layer of fluid over terrain. The form of the equations are:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{S}$$

where $\mathbf{U} = [h\ uh\ vh]^T$ is the vector of conserved quantities; water depth $h$, $x$-momentum $uh$ and $y$-momentum $vh$. Other quantities entering the system are bed elevation $z$ and stage (absolute water level) $w$, where $w = z + h$. The fluxes in the $x$ and $y$ directions, $\mathbf{E}$ and $\mathbf{G}$ are given by

$$\mathbf{E} = \begin{bmatrix} uh \\ u^2h + gh^2/2 \\ uvh \end{bmatrix} \text{ and } \mathbf{G} = \begin{bmatrix} vh \\ vuh \\ v^2h + gh^2/2 \end{bmatrix}$$

and the source term (which includes gravity and friction) is given by

$$\mathbf{S} = \begin{bmatrix} 0 \\ -gh(z_x + S_{fx}) \\ -gh(z_y + S_{fy}) \end{bmatrix}$$

where $S_f$ is the bed friction. The friction term is modelled using Manning's resistance law

$$S_{fx} = \frac{u\eta^2\sqrt{u^2+v^2}}{h^{4/3}} \text{ and } S_{fy} = \frac{v\eta^2\sqrt{u^2+v^2}}{h^{4/3}}$$

in which $\eta$ is the Manning resistance coefficient. The model does not currently include consideration of kinematic viscosity or dispersion.

As demonstrated in our papers, Zoppou and Roberts (1999) and Nielsen et al. (2005) these equations and their implementation in ANUGA provide a reliable model of general flows associated with inundation such as dam breaks, flooding and tsunamis.

## 3 FINITE VOLUME METHOD

We use a finite-volume method for solving the shallow water wave equations Zoppou and Roberts (1999). The study area is represented by a mesh of triangular cells as in Figure 1 in which the conserved quantities of water depth $h$, and horizontal momentum $(uh, vh)$, in each volume are to be determined. The size of the triangles may be varied within the mesh to allow greater resolution in regions of particular interest.

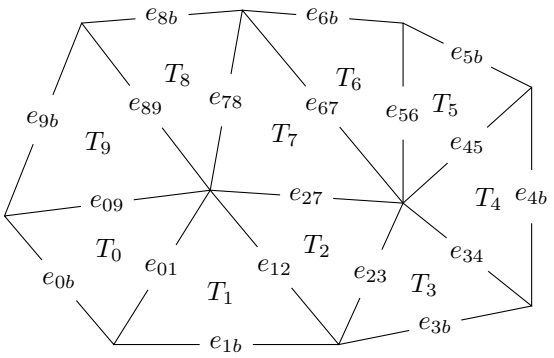


**Figure 1**. Triangular mesh used in our finite volume method. Conserved quantities $h$, $uh$ and $vh$ are associated with the centroid of each triangular cell, $T_i$. From the values of the conserved quantities at the centroid of a cell and its neighbouring cells, a discontinuous piecewise linear reconstruction of the conserved quantities is obtained. Fluxes are calculated across each edge $e_{ij}$ using the reconstructed quantities.

The equations constituting the finite-volume method are obtained by integrating the differential conservation equations over each triangular cell of the mesh.

Introducing some notation we use $i$ to refer to the $i$-th triangular cell $T_i$, and $N_i$ to the set of indices $j$ referring to cells neighbouring the $i$-th cell. Also let $A_i$ be the area of the $i$-th triangular cell, $e_{ij}$ the edge between the $i$-th and $j$-th cells, and $l_{ij}$ the length of the edge $e_{ij}$. Then integrating

$$\int_{T_i} \frac{\partial \mathbf{U}}{\partial t}\, d\mathbf{x} + \int_{T_i} \left( \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} \right)\, d\mathbf{x} = \int_{T_i} \mathbf{S}\, d\mathbf{x}$$

By an application of the divergence theorem, we obtain an equation for each cell $T_i$ which describes the rate of change of the average of the conserved quantities within each cell, in terms of the fluxes across the edges of the cells and the effect of the source terms.

$$\frac{d}{dt}\frac{1}{A_i}\int_{T_i} \mathbf{U}\, d\mathbf{x} + \frac{1}{A_i}\sum_{j\in N_i}\int_{e_{ij}} (\mathbf{E}, \mathbf{G})\cdot \mathbf{n}\, ds = \frac{1}{A_i}\int_{T_i} \mathbf{S}\, d\mathbf{x}$$

So the rate equations associated with average of each cell have the form

$$\frac{d\mathbf{U}_i}{dt} + \frac{1}{A_i}\sum_{j\in N_i} \mathbf{H}_{ij} l_{ij} = \mathbf{S}_i$$

where

- $\mathbf{U}_i$ the vector of conserved quantities averaged over the $i$th cell, $\frac{1}{A_i}\int_{T_i} \mathbf{U}\, d\mathbf{x}$,
- $\mathbf{S}_i$ is the average of the source term associated with the $i$th cell, $\frac{1}{A_i}\int_{T_i} \mathbf{S}\, d\mathbf{x}$,
- $l_{ij}$ is the length of the edge $e_{ij}$, and

- $\mathbf{H}_{ij} l_{ij}$ an approximation of the outward normal flux of material across the *ij*th edge, $\int_{e_{ij}} (\mathbf{E}, \mathbf{G}) \cdot \mathbf{n}\, ds$.

Given the cell average values $\mathbf{U}_k$ (which are essentially centroid values), the calculation of the fluxes at the edges need an approximation of the conserved values at the edges. This is called the reconstruction process. It is common to use a second order reconstruction to produce a piece-wise linear reconstruction of the conserved quantities for all $x \in T_i$ for each cell. The slope of this function is limited to avoid artificially introduced oscillations. This function is allowed to be discontinuous across the edges of the cells. The reconstructed quantities in the $i$-th cell is denoted $\mathbf{U}_i(\mathbf{x})$. The values on either side of an edge are denoted

$$\mathbf{U}_{ij}^i = \lim_{\mathbf{x} \to m_{ij}} \mathbf{U}_i(\mathbf{x}) \quad \text{and} \quad \mathbf{U}_{ij}^j = \lim_{\mathbf{x} \to m_{ij}} \mathbf{U}_j(\mathbf{x}).$$

The flux in the direction $\mathbf{n}$, is given by

$$\mathbf{H}(\mathbf{U}) = \mathbf{E}(\mathbf{U}) n_1 + \mathbf{G}(\mathbf{U}) n_2.$$

The numerical scheme is obtained by approximating the flux $\mathbf{H}$ using a numerical flux function. Godunov's method (see Toro (1992)) involves calculating the numerical flux function by exactly solving the corresponding one dimensional Riemann problem normal to the edge. A much simpler approximation is the central-upwind scheme of Kurganov et al. (2000) which is given by

$$\mathbf{H}_{ij} = \frac{a_{ij}^+ \mathbf{H}(\mathbf{U}_{ij}^i) - a_{ij}^- \mathbf{H}(\mathbf{U}_{ij}^j)}{a_{ij}^+ - a_{ij}^-} + \frac{a_{ij}^+ a_{ij}^-}{a_{ij}^+ - a_{ij}^-} \left[ \mathbf{U}_{ij}^j - \mathbf{U}_{ij}^i \right]$$

with bounds on wave speeds given by

$$a_{ij}^+ = \max \left\{ u_{ij}^i + \sqrt{g h_{ij}^i}, u_{ij}^j + \sqrt{g h_{ij}^j}, 0 \right\}, \quad a_{ij}^- = \min \left\{ u_{ij}^i - \sqrt{g h_{ij}^i}, u_{ij}^j - \sqrt{g h_{ij}^j}, 0 \right\}.$$

An explicit Euler timestepping method with variable timestepping adapted to the observed CFL condition is used to approximate the semi-discrete equation. In particular

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{A_i} \sum_{j \in N_i} \mathbf{H}_{ij} l_{ij} + \Delta t\, \mathbf{S}_i$$

For stability of the method, it is necessary that the timestep satisfy the Courant-Fredrichs-Levy (CFL) condition $\Delta t \leq \min_i \min_{j \in N_i} \left( \frac{r_i}{a_{ij}}, \frac{r_j}{a_{ij}} \right)$ where $a_{ij} = \max\{a_{ij}^+, -a_{ij}^-\}$ and $r_i$ is the radius of the inscribed circle of the $i$-th cell.

## 4 PARALLEL IMPLEMENTATION

Currently ANUGA uses a simple domain decomposition technique to parallelize the code. The first step is to subdivide the mesh into, roughly, equally sized partitions. On a rectangular mesh this may be done by a simple co-ordinate based dissection method, but on complicated domains a more sophisticated approach must be used. We use a PYTHON wrapper around the METIS (`http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`) partitioning library. The code uses METIS to divide the mesh for parallel computation. METIS was chosen as the partitioner based on the results in the paper Karypis and Kumar (1999). To minimize the amount of communication, it is common to add an extra layer of cells, which we call ghost cells which hold any extra information that a processor may need to complete its calculations. The ghost cell values are updated through buffered communication calls. Figure 2 shows a partition of a mesh into two submeshes with the extra layer of ghost cells. When partitioning the mesh we introduce new, dummy, boundary edges. These new boundary edges are tagged as standard boundary edges.

A typical timestep consists of the following computational steps:

```
compute_fluxes()  # Compute  H_ij across each edge
compute_forcing_terms() # Compute S_i
update_timestep() # Communication a global timestep
update_conserved_quantities()
update_ghosts()   # Communicate cell values to neighbouring processors
```

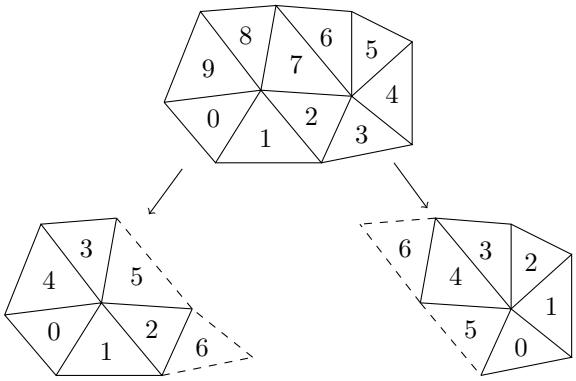


**Figure 2**. An example subpartitioning of original mesh, together with ghost triangles. The numbers show the local numbering scheme of the meshes.

The `compute_fluxes()` step is the most computationally intensive, consisting of iterating through all the cells or edges and accumulating, for each cell, the fluxes along each edge. This step itself consist of a number of calls to PYTHON routines. This produces an overhead which will be investigated in future. In addition there is great opportunity to improve the speed of this section of the code via the use of auxiliary computational units such as GPU's. The use of GPU's is non trivial and involves an extensive rewriting of the inner computational kernels and careful consideration on the placement and movement of data. But our initial experiments in this direction have demonstrated speedups of 20 times using GPUs.

There are two communications steps,

(1) `update_timestep()` which involves a `MPI_Allreduce` to find a global timestep for the scheme based on a CFL condition,

(2) `update_ghosts()` where at present we use the asynchronous MPI calls, where we initiate all the `MPI_Irecv` first and then the corresponding `MPI_Isend` calls are made. Finally the communication is resolved via a call to `MPI_Waitall`.

The use of these asynchronous MPI calls produced a great improvement in scalability of our code, jumping from modest results for 32 - 64 MPI processes up to 90% efficiency for larger problems with 100s to 1000s of MPI processes. There is still an opportunity to improve efficiency by overloading communication in `update_ghosts()` with computation in `compute_fluxes()` which we will investigate in future.

In Figure 3 we present experiments demonstrating the typical speedups for problems running from 43,200 triangles to 2,349,353 triangles. The results demonstrate super scalability for the larger problems stemming from better cache utilization for those problems based on the simple observation that the sub divided meshes can fit in the cache in those cases.

These experiments were conducted on a Fujitsu PRIMERGY cluster consisting of 36 BX920 S2 blades, each blade consisting of 2 Intel Xeon X5670 CPUs (Westmere, 6 core, 2.934 GHz) for a total of 432 cores.

## 5 CONCLUSIONS

We have presented our initial experiments using ANUGA to solve tsunami simulations involving millions of triangles using hundreds of MPI processes.

There is still work to be done to obtain better results on multi-thousand processor jobs. We have identified two important bottlenecks that need to addressed:

(1) The initial construction of the grid is a sequential task which due to memory restrictions usually has to be done on a separate large memory machine. We would like to undertake this stage in parallel. For instance recent work Chrisochoides et al. (2009) and Foteinos and Chrisochoides (2012) demonstrate the possibility of efficient parallel triangulation and partitioning. We will investigate the possibility of using similar techniques in ANUGA .

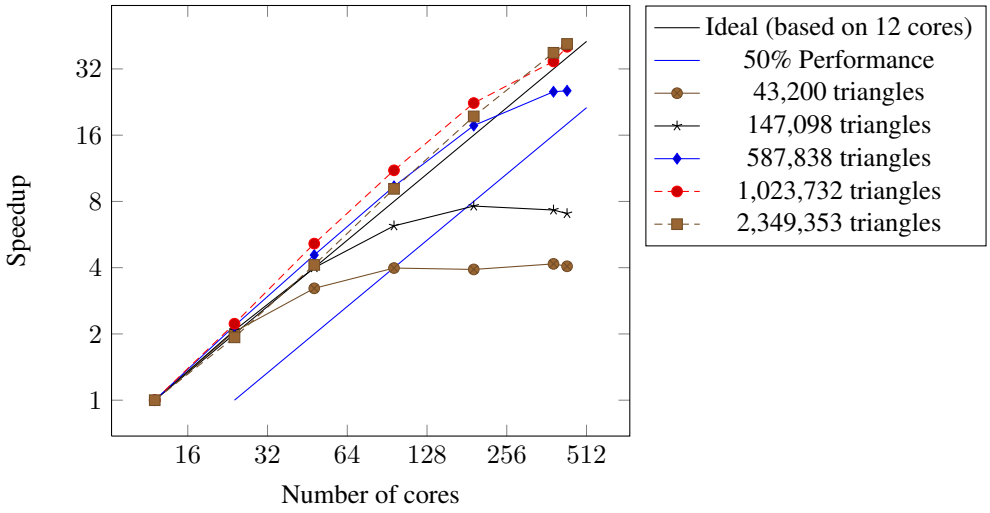


**Figure 3**. Speedups for problems ranging from 43,200 triangles to 2,349,353 triangles showing excellent speedups for the larger problems. We surmise that the super-scalability stems from better cache utilization as the larger meshes are subdivided into smaller meshes that can fit into cache.

(2) The other bottleneck is in the `compute_fluxes` part of the computation where most of the time (for multi-thousand processor jobs) is attributed to calling `C` code from PYTHON. A simple solution to this problem will be to implement more of the `evolve` loop in `C`.

In subsequent papers we will report on our efforts to address these bottlenecks and report on results of our simulations. Our ultimate goal is the solution of high (tens of millions of triangles) resolution flood and tsunami simulations in better than real time.

## ACKNOWLEDGEMENT

This work was supported by the Australian Research Council (ARC) and Fujitsu Laboratories of Europe (FLE) through the ARC National Competitive Grants Program (NCGP) Linkage Project LP110200410.

## REFERENCES

Begnudelli, L. and B. Sanders (2007). Simulation of the St. Francis dam-break flood. *Journal of Engineering Mechanics 133*, 1200.

Chippada, S., C. Dawson, M. Martínez, and M. Wheeler (1998). A Godunov-type finite volume method for the system of shallow water equations. *Computer methods in applied mechanics and engineering 151*(1-2), 105–129.

Chrisochoides, N., A. Chernikov, A. Fedorov, A. Kot, L. Linardakis, and P. Foteinos (2009). Towards exascale parallel delaunay mesh generation. In B. Clark (Ed.), *Proceedings of the 18th International Meshing Roundtable*, pp. 319–336. Springer Berlin Heidelberg.

Enkovaara, J., M. Louhivuori, P. Jovanovic, V. Slavnic, and R. Mikael (2011). Optimizing GPAW. Technical report. Available at `http://www.prace-ri.eu/IMG/pdf/Optimizing_GPAW.pdf`.

Foteinos, P. and N. Chrisochoides (2012). Dynamic parallel 3d delaunay triangulation. In W. Quadros (Ed.), *Proceedings of the 20th International Meshing Roundtable*, pp. 3–20. Springer Berlin Heidelberg.

Karypis, G. and V. Kumar (1999). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing 20*(1), 359.

Kurganov, A., S. Noelle, and G. Petrova (2000). Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton–Jacobi equations. *J. Comput. Phys 160*, 720–742.

Nielsen, O., S. Roberts, D. Gray, A. McPherson, and A. Hitchman (2005). Hydrodynamic modelling of coastal inundation. MODSIM 2005 International Congress on Modelling and Simulation, Modelling and Simulation Society of Australian and New Zealand, 518–523.

Toro, E. (1992). Riemann problems and the WAF method for solving the two-dimensional shallow water equations. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences 338*(1649), 43.

Zoppou, C. and S. Roberts (1999). Catastrophic collapse of water supply reservoirs in urban areas. *Journal of Hydraulic Engineering 125*(7), 686–695.