Parallelisation of a finite volume method for hydrodynamic inundation modelling

S. G. Roberts[1] L. Stals[2] O. M. Nielsen[3]

(Received 14 September 2006; revised 27 November 2007)

Abstract

Geoscience Australia and the Australian National University are developing a hydrodynamic inundation modelling tool called ANUGA to help simulate the impact of natural inundation hazards such as riverine flooding, storm surges and tsunamis. The core of ANUGA is a Python implementation of a finite volume method, based on triangular meshes, for solving the conservative form of the Shallow Water Wave equation. We describe the parallelisation of the code using a domain decomposition strategy where we use the METIS graph partitioning library to decompose the finite volume meshes. The parallel efficiency of our code is tested using a number of mesh partitions, and we verify that the METIS graph partition is particularly efficient.

See http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/153 for this article, Published November 27, 2007. ISSN 1446-8735

Contents

1 Introduction

Hydrodynamic modelling allows flooding, storm surge and tsunami hazards to be better understood, their impacts to be anticipated and, with appropriate planning, their effects to be mitigated. Geoscience Australia in collaboration with the Mathematical Sciences Institute, Australian National University, is developing a software application called ANUGA to model the hydrodynamics of floods, storm surges and tsunamis. These hazards are modelled using the conservative shallow water equations which are described in section 2. In ANUGA the solution of these equations are approximated using a finite volume method based on triangular meshes, as described in section 3. Nielsen et al. [4] provides a more complete discussion of the method and

also provides a validation of the model and method on a standard tsunami benchmark data set. Section 4 provides a description of the parallelisation of the code using a domain decomposition strategy and in section 5 preliminary timing results verify the scalability of the parallel code.

2 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 $\mathbf{U} = \begin{bmatrix} h & uh & vh \end{bmatrix}^T$ is the vector of conserved quantities: water depth h, and horizontal momentum $(u, v)h$. Other quantities entering the system are bed elevation z and stage w (absolute water level), where these variables are related via $w = z + h$. The horizontal fluxes in the x and y directions are

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

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

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

where S_f is the bed friction. We model the friction term using Manning's resistance law

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

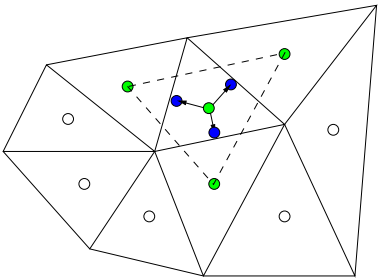

FIGURE 1: Conserved quantities h, uh and vh are associated with the centroid of each triangular cell. From the values of the conserved quantities at the centroid of the cell and its neighbouring cells, a discontinuous piecewise linear reconstruction of the conserved quantities is obtained.

in which η is the Manning resistance coefficient.

As demonstrated by Zoppou and Roberts [5], and Nielsen et al. [4], these equations provide an excellent model of flows associated with inundation such as dam breaks and tsunamis.

3 Finite volume method

We use a finite volume method for approximating the solution of the shallow water wave equations [5]. The study area is represented by a mesh of triangular cells as in Figure 1, in which the conserved quantities h, uh, and vh, in each cell are to be determined. The size of the triangular cells are varied within the mesh to allow greater resolution in regions of particular interest.

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 index of the ith triangular cell T_i, and $\mathcal{N}(i)$ to the set of indices referring to the cells neighbouring the ith cell. The area of the ith triangular cell is A_i and l_{ij} is the length of the edge between the ith and jth cells.

By applying the divergence theorem we obtain for each cell, an equation 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 cell and the effect of the source term. In particular, for each cell

$$\frac{d\mathbf{U}_i}{dt} + \frac{1}{A_i} \sum_{j \in \mathcal{N}(i)} \mathbf{H}_{ij} l_{ij} = \mathbf{S}_i \,,$$

where $\mathbf{U}_i$ is the vector of conserved quantities averaged over the ith cell, $\mathbf{S}_i$ is the source term associated with the ith cell and $\mathbf{H}_{ij}$ is the outward normal flux of material across the ijth edge.

The flux $\mathbf{H}_{ij}$ is evaluated using a numerical flux function $\mathbf{H}(\cdot, \cdot\,; \cdot)$ which is consistent with the shallow water flux in the sense that for all conserved quantity vectors $\mathbf{U}$ and spatial vectors $\mathbf{n}$

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

Then the flux across the ijth edge is

$$\mathbf{H}_{ij} = \mathbf{H}(\bar{\mathbf{U}}_i(m_{ij}), \bar{\mathbf{U}}_j(m_{ij}); \mathbf{n}_{ij}) \,,$$

where m_{ij} is the midpoint of the ijth edge and $\mathbf{n}_{ij}$ is the outward pointing normal of the ijth edge. The function $\bar{\mathbf{U}}_i(x, y)$ for $(x, y) \in T_i$ is obtained from the average conserved quantity values, $\mathbf{U}_k$, for $k \in \{i\} \cup \mathcal{N}(i)$ (ith cell and its neighbours). We use a second order reconstruction to produce a piecewise linear function reconstruction, $\bar{\mathbf{U}}_i(x, y)$, of the conserved quantities for all $(x, y) \in T_i$ for each cell (see Figure 1). This function may be discontinuous across the edges of the cells, but the slope of this function is limited to avoid artificially introduced oscillations.

The numerical flux function $\mathbf{H}(\cdot, \cdot\,; \cdot)$ is obtained by rotating the coordinate system so the outward normal is in the x direction. This then reduces the normal flux calculation to a one dimensional flux calculation. The central upwind scheme as described by Kurganov et al. [3] then approximates the resulting fluxes of the one dimension problem.

In the computations presented we use an explicit Euler time stepping method with variable time stepping adapted to the Courant Friedrichs Levy condition number.

4 Parallel implementation

To parallelise our algorithm we use a domain decomposition strategy. We start with a global mesh which defines the domain of our problem. We must first subdivide the global mesh into a set of submeshes. This partitioning is done using the METIS partitioning library [1]. We demonstrate the efficiency of this library in the following subsections. Once this partitioning has been done, a layer of 'ghost' cells is constructed and the corresponding communication patterns are set. Then we start to evolve our solution. The computation progresses independently on each submesh, until the end of the time step when the appropriate information is communicated among processing elements.

4.1 Subdividing the global mesh

The first step in parallelising the code is to subdivide the mesh into, roughly, equally sized partitions to ensure good load balancing. On a rectangular mesh this may be done by a simple coordinate based dissection method, but on a complicated domain such as the mesh shown in Figure 2, a more sophisticated approach must be used. We use the METIS partitioning library

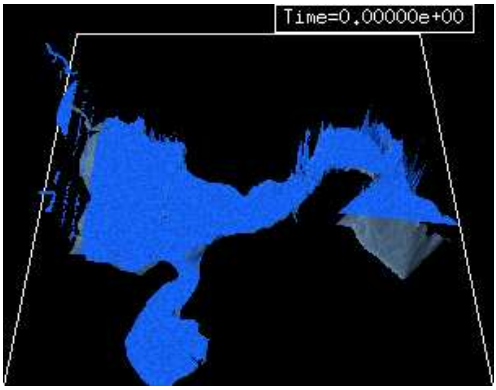

FIGURE 2: The global Lake Merimbula mesh

TABLE 1: 4-way and 8-way partition tests of Merimbula mesh showing the percentage distribution of cells between the submeshes.

CPU	0	1	2	3
Cells	2757	2713	2761	2554
%	25.6%	25.2%	25.6%	23.7%

CPU	0	1	2	3	4	5	6	7
Cells	1229	1293	1352	1341	1349	1401	1413	1407
%	11.4%	12.0%	12.5%	12.4%	12.5%	13.0%	13.1%	13.1%

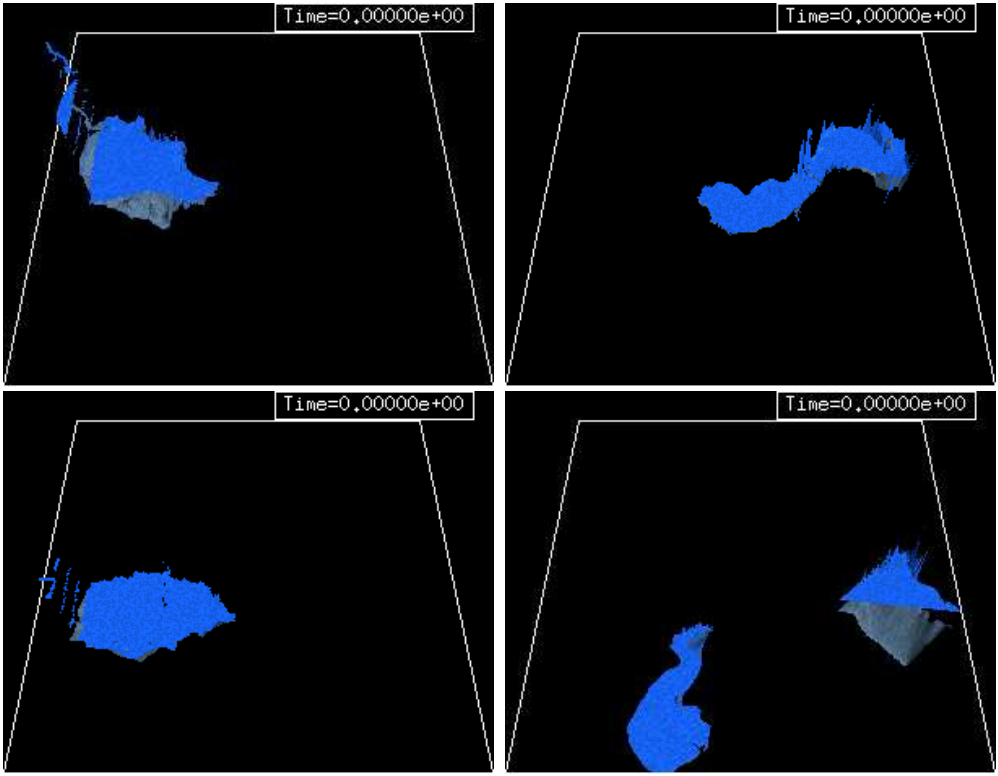

FIGURE 3: The global Lake Merimbula mesh from Figure 2, partitioned into four submeshes using METIS.

based on Karypis and Kumar's [2] recommendations. Figure 3 shows the original global grid partitioned over four submeshes. Note that one submesh may comprise several unconnected mesh partitions and Table 1 gives the node distribution over four submeshes and eight submeshes. These results imply that METIS gives a reasonably well balanced partition of the mesh.

Ghost cells Consider the example subpartitioning given in Figure 4. The top mesh represents the global mesh, whereas the two lower meshes display the partitioning of the global mesh (together with extra ghost cells to facilitate communication) onto two processors.

As an example, during the evolution calculations, cell 2 (residing on processor 0) will need to access information from its global neighbour, cell 3 (residing on processor 1). A standard approach to this problem is to add an extra layer of cells, which we call ghost cells, to each submesh, on each processor. The ghost cells are used to hold information that an ordinary cell in a submesh needs to complete its calculations. The values associated with the ghost cells need to be updated through communication calls usually only once every time step (at least for first order time stepping). Such communication patterns are determined and recorded before sub partitioning the mesh into submeshes.

The ghost cells in each submesh are treated exactly the same as any other cell, the only way to differentiate them is to look at the communication pattern. This means that the code is essentially the same whether it is being run in serial or parallel, the only difference is a communication call at the end of each time step and an extra `if` statement in the local calculation of the time step constraint to ensure that the ghost cells are not used in that calculation.

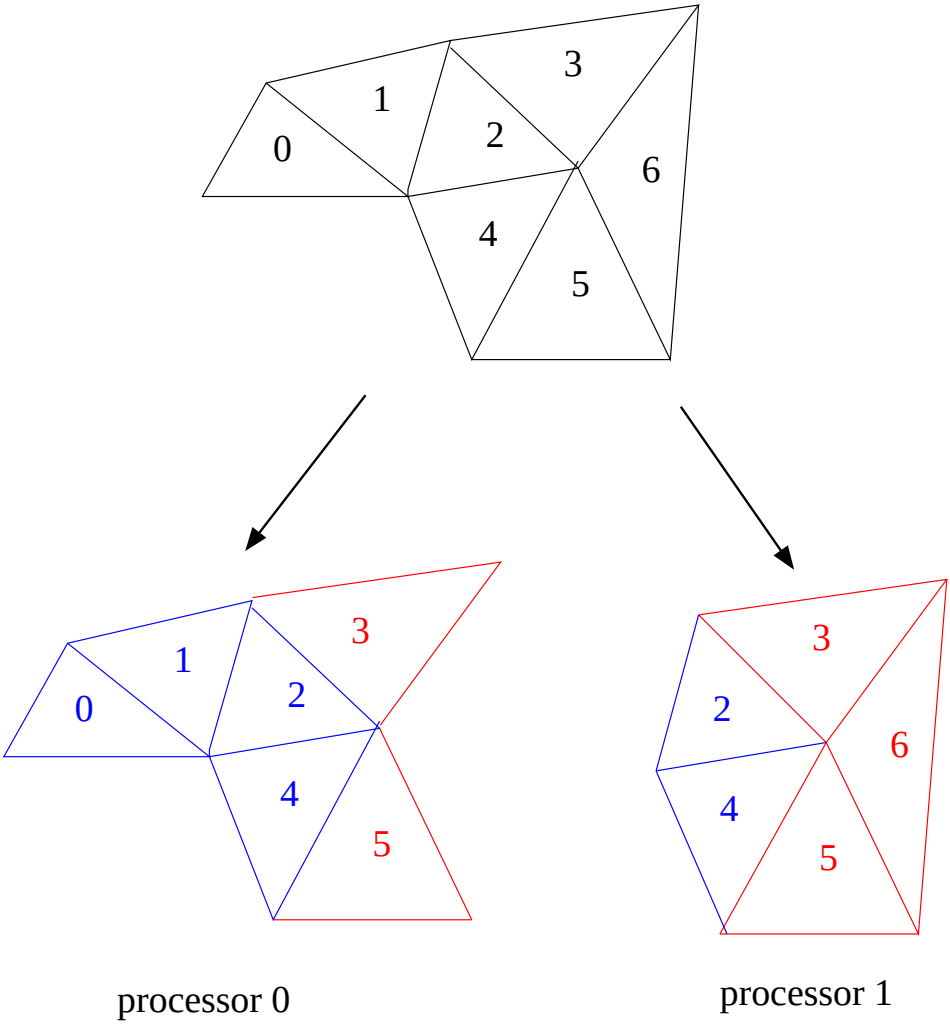

FIGURE 4: An example subpartitioning of a global mesh into two submeshes, showing the ghost nodes used in the two submeshes.

5 Performance analysis

To evaluate the performance of the code on a parallel machine we ran some examples on a cluster of four nodes connected with PathScale InfiniPath HTX. Each node has two AMD Opteron 275 (Dual core 2.2 GHz Processors) and 4 GB of main memory. The system achieves 60 Gigaflops with the Linpack benchmark, which is about 85% of peak performance. For each test run we evaluate the parallel efficiency as

$$E_p = \frac{T_1}{pT_p} 100\,,$$

where $T_p = \max_{0 \le i < p}\{t_i\}$, p is the total number of processors and t_i is the time required to run the evolution code on processor i. Note that t_i only includes the time required to do the finite volume evolution calculations, not the time required to build and subpartition the mesh.

5.1 Advection, rectangular mesh

The first example solves an advection equation on a rectangular mesh of size n by m. Table 2 show the efficiency results for different values of n and m. The examples where $p \le 4$ were run on one Opteron node containing four processors, the $p = 8$ example was run on two nodes (giving a total of eight processors). The communication within a node is faster than the communication across nodes, so we would expect to see a decrease in efficiency when we jump from four to eight processors. On the other hand, the efficiency is likely to increase with n and m, due to an increased ratio between interior and exterior triangles and hence an increased ratio of computation to communication. The results displayed in Table 2 verify this, except perhaps for the slightly higher than expected efficiency of the $p = 2$, $n = 80$, $m = 80$ case. Generally the efficiency results shown here are consistent and competitive.

TABLE 2: Parallel efficiency results for the advection problem on n by m rectangular meshes using p processors.

	40 by 40 mesh		80 by 80 mesh		160 by 160 mesh	
p	T_p (sec)	$E_p(\%)$	T_p (sec)	$E_p(\%)$	T_p (sec)	$E_p(\%)$
1	36.6		282.		2200.	
2	18.8	98	143.	99	1126.	98
4	10.2	90	75.1	94	569.	97
8	6.39	72	41.7	85	304.	90

TABLE 3: Parallel efficiency results for the advection equation and the shallow water equation on the Lake Merimbula mesh for p processors.

	Advection		Shallow water eqn	
p	T_p (sec)	$E_p(\%)$	T_p (sec)	$E_p(\%)$
1	145.0		7.04	
2	77.5	94	3.62	97
4	41.2	88	1.94	91
8	23.0	79	1.15	76

5.2 Advection, Lake Merimbula mesh

We now look at another advection example where the mesh comes from a study of water flow in Lake Merimbula, New South Wales. The mesh is shown in Figure 2. The results are given in Table 3. These are good efficiency results, especially considering the structure of this mesh.

5.3 Shallow water, Lake Merimbula mesh

The final example we look at is the shallow water equation on the Lake Merimbula mesh. The results for this case are also listed in Table 3. Efficiency for two and four processors is again good. For eight processors the efficiency falls off rather quickly.

On profiling the code we found that the loss of efficiency is due to the boundary update routine. To allow maximum flexibility in experimenting with different boundary conditions, the boundary routines are written in PYTHON (as opposed to most of the other computationally intensive kernels which are written in C). When running the code on one processor the boundary routine accounts for about 72% of the total computation time. The METIS subpartition of the mesh produced an imbalance in the number of active boundary edges in each subpartition. The profiler indicated that when running the problem on eight processors, Processor 0 spent about 3.8 times more time on the boundary calculation than Processor 7, indicating about 3.8 times as many active boundary edges. This load imbalance reduced the parallel efficiency. In the future the boundary routine will be rewritten in C to reduce its overall contribution to the computation and so reduce the effect of this active boundary imbalance.

6 Conclusions

ANUGA is a flexible and robust modelling system that simulates hydrodynamics by solving the shallow water wave equation in a triangular mesh. It models the process of wetting and drying as water enters and leaves an area and is capable of capturing hydraulic shocks due to the ability of the finite volume method to accommodate discontinuities in the solution. It simulates the behaviour of hydrodynamic natural hazards such as riverine flooding, storm surge and tsunami.

The use of the parallel code will enhance the modelling capability of ANUGA and will form part of Geoscience Australia's ongoing research effort to model and understand the potential impact from natural hazards in order to reduce their impact on Australian communities.

References

[1] George Karypis. METIS—serial graph partitioning and fill-reducing matrix ordering. http://glaros.dtc.umn.edu/gkhome/metis/metis/overview, 2006. C563

[2] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999. doi:10.1137/S1064827595287997 C566

[3] A. Kurganov, S. Noelle, and G. Petrova. Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton–Jacobi equations. *SIAM Journal of Scientific Computing*, 23(3):707–740, 2001. doi:10.1137/S1064827500373413 C563

[4] O. Nielsen, S. Roberts, D. Gray, A. McPherson, and A. Hitchman. Hydrodynamic modelling of coastal inundation. In A. Zerger and R.M. Argent, editors, *MODSIM 2005 International Congress on Modelling and Simulation*, pages 518–523. Modelling and Simulation Society of Australia and New Zealand, December 2005. http://www.mssanz.org.au/modsim05/papers/nielsen.pdf. C559, C561

[5] C. Zoppou and S. Roberts. Catastrophic Collapse of Water Supply Reservoirs in Urban Areas. *ASCE J. Hydraulic Engineering*, 125(7):686–695, 1999. doi:10.1061/(ASCE)0733-9429(1999)125:7(686) C561

Author addresses

1. **S. G. Roberts**, Dept. of Maths, Australian National University, Canberra, AUSTRALIA.
mailto:stephen.roberts@anu.edu.au

2. **L. Stals**, Dept. of Maths, Australian National University, Canberra, AUSTRALIA.
mailto:linda.stals@anu.edu.au

3. **O. M. Nielsen**, Risk Assessment Methods Project, Geospatial and Earth Monitoring Division, Geoscience Australia, Symonston, AUSTRALIA.
mailto:Ole.Nielsen@ga.gov.au