A Parallel High-Order Discontinuous Galerkin Shallow Water Model

Claes Eskilsson[1], Yaakoub El-Khamra[1], David Rideout[2], Gabrielle Allen[1], Q. Jim Chen[1], and Mayank Tyagi[1]

[1] Louisiana State University, Baton Rouge LA 70803, USA
[2] Perimeter Institute for Theoretical Physics, Waterloo Ontario N2L 2Y5, Canada

Abstract. The depth-integrated shallow water equations are frequently used for simulating geophysical flows, such as storm-surges, tsunamis and river flooding. In this paper a parallel shallow water solver using an unstructured high-order discontinuous Galerkin method is presented. The spatial discretization of the model is based on the Nektar++ spectral/*hp* library and the model is numerically shown to exhibit the expected exponential convergence. The parallelism of the model has been achieved within the Cactus Framework. The model has so far been executed successfully on up to 128 cores and it is shown that both weak and strong scaling are largely independent of the spatial order of the scheme. Results are also presented for the wave flume interaction with five upright cylinders.

1 Introduction

This paper presents a first step towards a community coastal modeling toolkit for nearshore surface water waves based on spectral/*hp* element methods. The ultimate goal is a scalable, parallel, non-hydrostatic wave solver, based on multi-layered Boussinesq-type equations including time-dependent bathymetry and sediment transport. In this paper we outline the ongoing work of the parallel implementation of non-dispersive two-dimensional shallow water equations (SWE). As Boussinesq-type equations are higher-order extensions to the SWE, the SWE constitute the natural initial stepping stone. A driving force behind this effort is storm surge modeling and the SWE model presented here can be used as an efficient hydrodynamic core for such simulations.

There are several motivations for using spectral/*hp* element methods (i.e. finite element methods of arbitrarily high order) rather than more traditional methods. Spectral/*hp* elements provide a flexible setting where both the element size and the polynomial order within the elements can be altered. Further, high-order methods tend to be more computationally efficient for long-time integrations compared to low-order methods due to their inherently small numerical diffusion/dispersion error. The discontinuous Galerkin (DG) flavour of spectral/*hp* elements was choosen mainly since the resulting global mass matrix is block-diagonal. Thus, any explicit time-stepping scheme can update the

G. Allen et al. (Eds.): ICCS 2009, Part I, LNCS 5544, pp. 63–72, 2009.

numerical solution in an element-by-element fashion. The DG method is also able to incorporate well established shock-capturing techniques from the finite volume (FV) framework.

SWE models are most frequently based on shock-capturing FV methods, but over the last decade several DG SWE models have been presented. Later studies emphasize the use of high-order schmes [5,4,11,6] and the present state-of-the-art DG SWE models rely on adaptivity: [1] include h-type adaptivity while [13] illustrates the benefits of p-type adaptivity. In contrast to a recent study [12] that addresses the parallel performance of a low-order DG SWE model, the present study is addressing the performance of high-order schemes.

The coastal DG wave model is based on the spectral/hp element library Nektar++ [9,10], while parallelism has been achieved by adding support for unstructured two-dimensional meshes into the computational framework Cactus [8,7]. This separation of programming tasks allows coastal engineers to focus on developing coastal code using Nektar++ and the computational scientists to focus on parallelism, performance and scalability of the unstructured mesh driver. Further, the adoption of an underlying parallel framework provides a methodology to develop interoperable modules to add additional solvers and models leading to a comprehensive toolkit for modeling coastal environments.

2 Shallow Water Equations

The shallow water equations are a set of non-linear hyperbolic equations. The equations are derived under the assumption of hydrostatic pressure, and thus the SWE are only valid for long waves (the rule of thumb being that the still water depth to wavelength ratio should be less than 1/20). The conservation form of the SWE read

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \mathbf{S}(\mathbf{U}) \,, \tag{1}$$

where $\mathbf{U} = [H \,, Hu \,, Hv]^{\mathrm{T}}$ is the vector of conserved variables and $\mathbf{F}(\mathbf{U}) = [\mathbf{E}(\mathbf{U}) \,, \mathbf{G}(\mathbf{U})]^{\mathrm{T}}$ is the flux vector defined by:

$$\mathbf{E} = \begin{bmatrix} Hu \\ Hu^2 + gH^2/2 \\ Huv \end{bmatrix} \,, \qquad \mathbf{G} = \begin{bmatrix} Hv \\ Huv \\ Hv^2 + gH^2/2 \end{bmatrix} \,.$$

Here $H(\mathbf{x}, t) = \eta(\mathbf{x}, t) + d(\mathbf{x})$ is the total water depth, $\eta(\mathbf{x}, t)$ is the free surface elevation and $d(\mathbf{x})$ is the still water depth; $\mathbf{u}(\mathbf{x}, t) = [u(\mathbf{x}, t) \,, v(\mathbf{x}, t)]^{\mathrm{T}}$ denotes the depth-averaged velocity in the x- and y-direction, respectively, while g is the acceleration of gravity.

The source terms $\mathbf{S}(\mathbf{U})$ can contain forcing due to e.g. bathymetry, bottom friction, atmospheric pressure, Coriolis force, wind stresses and diffusion. The objective of this study is to assess the performance of the computational core rather than present any solution of real-life engineering cases, and we focus here on the homogeneous version of the SWE.

3 Numerical Scheme

3.1 DG Discretization

Let Ω_h denote the partition of the computational domain, Ω, into N non-overlapping elements Ω_e with elemental boundaries Γ_e. The diameter of the element Ω_e is given by h_e and subsequently $h = \max(h_1, \ldots, h_N)$. We introduce the discrete polynomial space

$$\mathcal{V}_\delta = \left\{ v \in L^2(\Omega) \,:\, v|_{\Omega_e} \in \mathcal{P}^p(\Omega_e), \forall \Omega_e \in \Omega_h \right\} ,$$

where $\mathcal{P}^p$ is the space of polynomials of degree at most p in the element Ω_e.

We proceed by multiplying eq. (1) with a piecewise smooth test function $q(\mathbf{x})$ and integrate over the local element Ω_e. We then approximate $\mathbf{U}$ and q with polynomial expansions of order p:

$$\int_{\Omega_e} q_\delta \frac{\partial \mathbf{U}_\delta}{\partial t}\, d\mathbf{x} + \int_{\Omega_e} q_\delta \nabla \cdot \mathbf{F}(\mathbf{U}_\delta)\, d\mathbf{x} = \int_{\Omega_e} q_\delta\, \mathbf{S}(\mathbf{U}_\delta)\, d\mathbf{x} .$$

After applying the divergence theorem and exchanging the boundary flux term with a numerical flux, we can state the discrete DG method as: find $\mathbf{U}_\delta \in \mathcal{V}_\delta$ such that for all $q_\delta \in \mathcal{V}_\delta$ and for all $\Omega_e \in \Omega_h$

$$\begin{aligned} \int_{\Omega_e} q_\delta \frac{\partial \mathbf{U}_\delta}{\partial t}\, d\mathbf{x} - \int_{\Omega_e} \nabla q_\delta \cdot \mathbf{F}(\mathbf{U}_\delta)\, d\mathbf{x} \\ + \int_{\Gamma_e} q_\delta\, \hat{\mathbf{F}}(\mathbf{U}_\delta) \cdot \mathbf{n}\, dS = \int_{\Omega_e} q_\delta\, \mathbf{S}(\mathbf{U}_\delta)\, d\mathbf{x} , \end{aligned} \tag{2}$$

where $\mathbf{n}$ is the outward unit normal to Γ_e and $\hat{\mathbf{F}}$ denotes the continuous numerical flux used to couple the elements together. In this study we use the popular HLLC Riemann solver [14] as the numerical flux.

The approximation of an arbitrary variable $f_\delta \in \mathcal{V}_\delta$ in the local element Ω_e can be expressed as

$$f_\delta(\mathbf{x}, t) = \sum_{i=0}^{N^e_{\text{dof}}-1} \tilde{\mathbf{f}}_e[i] \phi_i(\mathbf{x}) , \qquad \mathbf{x} \in \Omega_e ,$$

where $\tilde{\mathbf{f}}_e[i]$ is the time-dependent vector consisting of the N^e_{dof} elemental degrees of freedom of expansion coefficients and $\phi_i(\mathbf{x})$ are the trial functions.

Introducing the elemental mass matrix

$$\mathbf{M}^e[p][q] = \int_{\Omega_e} \phi^e_p(\mathbf{x})\, \phi^e_q(\mathbf{x})\, d\mathbf{x} ,$$

and the elemental weak derivative matrices

$$\mathbf{D}^e_x[p][q] = \int_{\Omega_e} \frac{\partial \phi^e_p(\mathbf{x})}{\partial x} \phi^e_q(\mathbf{x})\, d\mathbf{x} , \qquad \mathbf{D}^e_y[p][q] = \int_{\Omega_e} \frac{\partial \phi^e_p(\mathbf{x})}{\partial y} \phi^e_q(\mathbf{x})\, d\mathbf{x} ,$$

where $0 \leq p, q \leq N_{\text{dof}}^e - 1$, we can write eq. (2) in elemental form as

$$\mathbf{M}^e \frac{\partial \tilde{\mathbf{U}}}{\partial t} - \mathbf{D}_x^e \, \mathbf{E}(\tilde{\mathbf{U}}) - \mathbf{D}_y^e \, \mathbf{G}(\tilde{\mathbf{U}}) + \mathbf{f}^e = \mathbf{M}^e \, \mathbf{S}(\tilde{\mathbf{U}}) \,,$$

in which

$$\mathbf{f}^e[p] = \int_{\Gamma_e} \phi_p(\mathbf{x}) \left(\hat{\mathbf{F}}(\mathbf{U}_\delta) \cdot \mathbf{n} \right) \, dS \,.$$

In contrast to continuous Galerkin methods, for DG methods the global matrices are just a concatenation of local elemental matrices. Hence, using explicit time-stepping schemes the solution can be computed in an element-by-element fashion.

Following the standard Galerkin formulation we use the same functions for the test and trial functions. Former studies concerned with high-order DG methods for SWE have primarily used either the modal orthogonal Prioli-Koornwinder-Dubiner (PKD) basis [4,11] or the nodal electrostatic basis [5,6]. Here we use the C^0 modified PKD basis [10], which can be decomposed into boundary and interior modes. For this choice of basis the elemental mass matrix is sparse, but on the other hand there are only the boundary modes involved in the computation of the boundary integral. More importantly this choice of expansion basis gives the possibility to use static condensation if a global equation system must be solved [10]. Admittedly, the present SWE solver does not involve a global solve, but this should be an important feature in the future development of a Boussinesq solver.

The semi-discretized equations are advanced in time using a second- or third-order TVD Runge-Kutta method and all boundary conditions are enforced weakly through the use of the Riemann solver.

3.2 Computational Approach

The discontinuous Galerkin scheme outlined in section 3.1 was first implemented in a stand-alone serial code using the open-source spectral/*hp* library Nektar++ [9]. Nektar++ provides the fundamental tools associated with a high-order finite element method, such as the calculation of expansion functions, inner products and differentiation. With regard to the high-order discretization our work has been focused on implementing a solver structure for time-dependent problems. This includes a SWE class containing functions for the evaluation of the flux vector, numerical fluxes, equation dependent boundary condition, various source terms, etc.

We rely on the Cactus Framework [8] to provide parallelization and to adopt an extensible component approach to code development. Cactus is an open source problem solving environment designed for scientists and engineers needing to develop collaborative code for large scale parallel machines. Cactus comprises sets of components (or *thorns*) which are invoked by the Cactus *Flesh* which collects information from the thorns to define grid variables, parameters, methods and scheduling. The modular structure of Cactus enables parallel computation

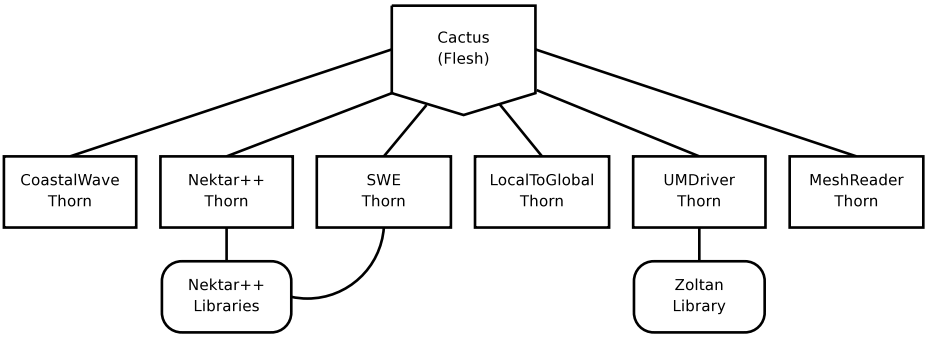

Fig. 1. Architectural diagram showing Cactus thorns and their connection to the external Nektar++ and Zoltan packages

across different architectures and collaborative code development between different groups.

To integrate the serial SWE solver into Cactus several new thorns were developed (see Figure 1). First, an unstructured mesh driver (thorn `UMDriver`) was developed which provides the underlying parallel layer and the Cactus Configuration Language was extended to support grid functions on unstructured meshes. The UMDriver (which is still under development) uses the Zoltan [3] library to provide mesh partitioning, load balancing and mesh migration. Another thorn `LocalToGlobal` provides local reindexing of elements, edges and vertices.

The core thorn for the coastal modeling toolkit in Cactus is `CoastalWave`. Mirroring the methodology for other domain specific toolkits in Cactus, this thorn defines the generic variables, parameters, and methods for coastal models, allowing models providing the same functionality to be swapped (e.g. the SWE will be interoperable with the Boussinesq solver) and allowing additional components to be added into the workflow (e.g. phase-averaging wave models to compute radiation stresses).

Thorn `Nektar++` initializes and populates the data structures of the Nektar++ library. Thorn `SWE` thorn contains the actual SWE solver based on routines defined in the SWE class. Finally, thorn `MeshReader` provides a simple ASCII mesh file reader, and also allows users to register their own mesh readers.

4 Computational Results

4.1 Convergence

Consider the simple case of a linear standing wave with a wavelength of 10 m in a square 10 by 10 m basin. The still water depth is 0.5 m. In order to compare with the analytical solution we here use the linearized SWE. We compute the solution for one wave period using 10 000 time steps. Table 1 shows the error and order of convergence measured in the L_2 norm.

Table 1. L_2 error and order of convergence for the linear standing wave

	$N = 16$	$N = 64$		$N = 256$	
p	error	error	order	error	order
1	3.3676E-03	6.4677E-04	2.38	1.6054E-04	2.01
2	1.7754E-04	2.1496E-05	3.05	2.5909E-06	3.05
3	2.1303E-05	2.2351E-06	3.25	6.3190E-08	5.14
4	3.4041E-07	9.7791E-09	5.12	3.0563E-10	5.00

In general we have optimal convergence $\mathcal{O}(h^{p+1})$, although for $p = 3$ there is an instance of $\mathcal{O}(h^p)$ convergence. This is due to the use of a simplified numerical flux (componentwise averaging) in the linear scheme. The use of averaging is known to sometimes produce sub-optimal convergence for odd p. Nevertheless, for p-type refinement we obtain the expected exponential convergence, illustrated by the approximate straight lines in Figure 2.

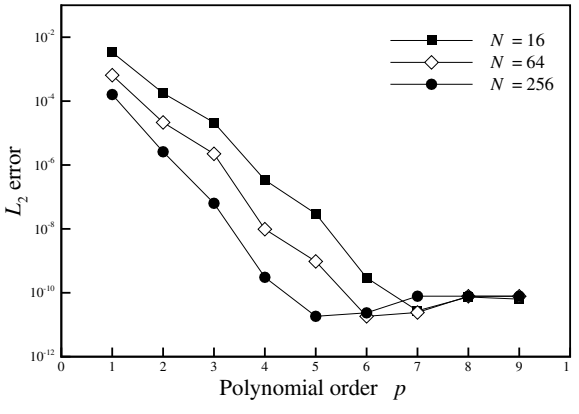

Fig. 2. Illustration of exponential convergence for the linear standing wave

4.2 Weak Scaling

Weak scaling indicates the ability of a code to scale up a problem on more cores, increasing the domain size or grid refinement while keeping a constant load on each core. We consider two series of meshes – consisting of 100 and 900 quadrilaterals per core, respectively. Both sets are run with three different polynomial orders: $p = 4, 6, 8$. The largest run (900 elements per core on 128 cores) thus contains 218 700 unique degrees of freedom per core, or a total of roughly 28 million degrees of freedom. We execute the model for one hundred time steps. Scaling tests were performed on the LONI "Queen Bee" Linux cluster (comprising 668 nodes, each containing dual Quad Core Xeon 64-bit 2.33GHz processors with an Infiniband interconnect).

Table 2. Weak scaling using 100 quadrilaterals [Top table] and 900 quadrilaterals [Bottom table] per core. Solution time is wall clock time in seconds while parallel efficiency is presented as an percentage.

Numbers	$p = 4$		$p = 6$		$p = 8$	
of cores	Solution time	Parallel eff.	Solution time	Parallel eff.	Solution time	Parallel eff.
1	5.00	—	7.01	—	10.6	—
2	5.56	89.8	7.77	90.3	11.7	90.6
4	6.16	81.2	8.54	82.1	12.7	83.4
8	7.17	69.7	9.98	70.3	14.7	72.1
16	7.53	66.3	10.35	67.8	15.3	69.0
32	7.75	64.5	10.63	66.0	16.0	66.1
64	8.44	59.2	11.45	61.3	16.5	64.0
128	9.06	55.2	12.13	57.8	17.2	61.4

Numbers	$p = 4$		$p = 6$		$p = 8$	
of cores	Solution time	Parallel eff.	Solution time	Parallel eff.	Solution time	Parallel eff.
1	134.3	—	153.1	—	185.7	—
2	140.6	95.6	160.1	95.6	193.7	95.8
4	150.5	89.3	169.5	90.3	203.5	91.3
8	159.8	84.1	181.9	84.1	218.6	85.0
16	167.4	80.3	185.9	82.3	222.5	83.5
32	165.0	81.4	190.0	80.6	226.0	82.2
64	167.1	80.4	194.5	78.7	227.8	81.5
128	170.6	78.7	191.4	80.0	237.6	78.2

Table 2 shows the solution time (without time spent in I/O, initializing and partitioning of the mesh) for the 100 and 900 element series. The parallel efficiency decreases for both series, although more rapidly for the 100 element series. This is due to the fact that ghost elements are currently used rather than ghost edges, so that the workload per core is not actually constant: a two-core partition has 100 plus an additional 10 ghost elements per core; four partitions has 100 elements plus 20 ghost elements per core, etc. This increase in computational overhead is especially pronounced for scaling tests with a low workload per core as well as for tests using less than eight cores. The parallel efficiency can be regarded as largely independent with regard to polynomial order p – the slight increase in efficiency for higher p can be attributed to the higher workload per core.

Scaling for the total execution time shows a larger decrease in efficiency, indicating that the initialization and mesh partitioning routines also require further attention to improve scalability. A core contributor to this effect is due to the fact that currently each core stores information about the indices on the entire mesh.

4.3 Strong Scaling

Strong scaling indicates the ability to decrease the total run time for a particular problem, scaling across more cores while keeping the overall problem size fixed.

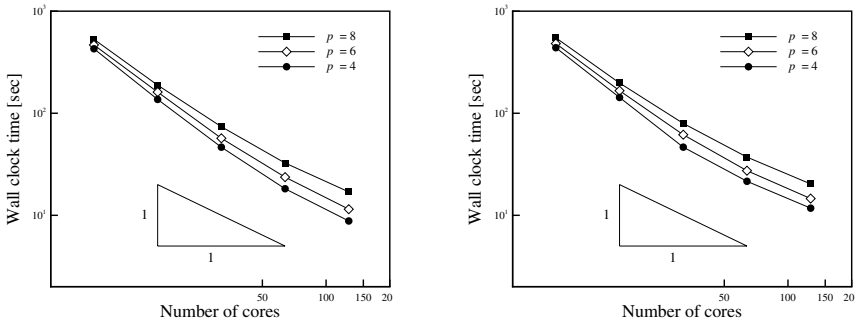

Fig. 3. Strong scaling. [Left] Solution time and [Right] total execution time.

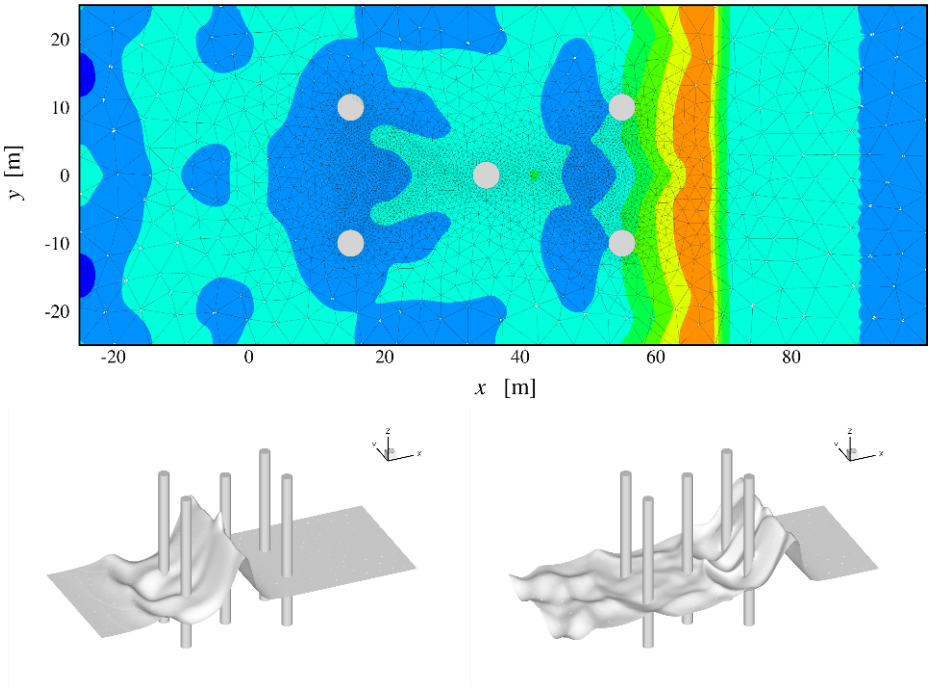

Fig. 4. Cylinder test case: [Top] computational mesh including iso-contours of the surface elevation at $t = 20$ seconds; [Bottom left] snapshot of surface elevation at $t = 9.9$ seconds and [Bottom right] snapshot of surface elevation at $t = 20$ seconds

We assess the strong scaling using a fixed size problem of 12 800 quadrilaterals. As for the weak scaling tests we use $p = 4\,,6\,,8$ and run the problem for 100 time steps. Figure 3 shows the reduction in solution time as the number of processors are increased. The curves in Figure 3 asymptotically approaches the 1:1 slope, indicating that the wall clock time is halved as the number of cores are doubled.

4.4 Wave Interacting with Cylinders

Consider a numerical wave flume that is 125 m long and 40 m wide, with an array of five cylinders (all having a 4 m diameter). The computational domain is discretized into roughly 6 000 triangular elements of polynomial order $p = 6$ (roughly 500 000 degrees of freedom), see the top image in Figure 4, heavily clustered around the cylinders. The initial condition is given by Laitone's first order solitary wave solution using an amplitude of 0.05 m centered at $x = 0$. We run the model for 25 seconds, using 10 000 steps. The non-linear SWE model is executed using 64 cores and the simulation generally takes 0.1 second per time step.

The lower images in Figure 4 shows the evolution of the solitary wave, including wave run-up, scattering, diffraction, reflection as well as interaction between the scattered waves. Note that the solution correctly remains symmetric throughout the simulation.

5 Concluding Remarks

We have outlined a spectral/*hp* DG method for solving the SWE and described it's integration with the Cactus Framework. This work concludes the first phase of a program to develop a scalable, parallel, non-hydrostatic wave solver, based on multi-layered Boussinesq-type equations capable of including time-dependent bathymetry and sediment transport. An important step has been taken in designing the integration of the underlying spectral/*hp* solver apparatus into Cactus to form the basis of a community framework for coastal modeling. This necessitated the development of an unstructured mesh driver for Cactus, which through the modular framework can now be used by other application domains.

Much work remains to be done. Comparing to the weak scaling results of e.g. [2] it is clear that the scaling of the SWE is not yet optimal. The main reasons for the suboptimal behaviour are: (i) additional communication overhead due to the use of complete ghost elements rather than just using ghost edges; (ii) current inefficiencies in the Cactus UMDriver thorn, e.g. in duplicating the physical space variables.

That the scaling was found independent of polynomial approximation is encouraging, as we can expect scalability also for higher-order schemes. Planned future work on the unstructured mesh driver includes improving scaling on large numbers of cores, adaptive mesh refinement, a hyperslabbing interface and dynamic load balancing. On the coastal modeling side, subroutines for e.g. flooding/drying and breaking waves will be added to allow for real-life engineering applications.

Acknowledgements

The authors gratefully acknowledge the contributions of Prof. S.J. Sherwin, Mr. P. Vos (Imperial College London) and Prof. R.M. Kirby (University of Utah) to the DG SWE solver. Financial support was provided by DOD/ONR

(N00014-07-1-0955), NSF/EPSCOR (EPS-0701491) and the Floating Point Systems Endowed Chair at LSU. This research was supported in part by the Perimeter Institute for Theoretical Physics. Computer resources were provided by the Louisiana Optical Network Initiative (LONI).

References

1. Bernard, P.-E., Chevaugeon, N., Legat, V., Deleersnijder, E., Remacle, J.-F.: High-order h-adaptive discontinuous Galerkin methods for ocean modelling. Ocean Dyn. 57, 109–121 (2007)
2. Biswas, R., Devine, K.D., Flaherty, J.: Parallel, adaptive finite element methods for conservation laws. Appl. Numer. Math. 14, 255–283 (1994)
3. Devine, K., Boman, E., Heaphy, R., Hendrickson, B., Vaughan, C.: Zoltan Data Management Services for Parallel Dynamic Applications. Comp. Sci. Eng. (2002)
4. Eskilsson, C., Sherwin, S.J.: A triangular spectral/hp discontinuous Galerkin method for modelling 2D shallow water equations. Int. J. Num. Meth. Fluids 45, 605–623 (2004)
5. Giraldo, F.X., Hesthaven, J.S., Warburton, T.: Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations. J. Comp. Phys. 181, 499–525 (2002)
6. Giraldo, F.X., Warburton, T.: A high-order triangular discontinuous Galerkin oceanic shallow water model. Int. J. Num. Meth. Fluids 56, 899–925 (2008)
7. Goodale, T., Allen, G., Lanfermann, G., Massó, J., Radke, T., Seidel, E., Shalf, J.: The Cactus framework and toolkit: Design and applications. Vector and Parallel Processing. Springer, Heidelberg (2003)
8. `http://www.cactuscode.org`
9. `http://www.nektar.info`
10. Karniadakis, G.E., Sherwin, S.J.: Spectral/hp Element Methods for Computational Fluid Dynamics. Oxford Sci. Publ. (2005)
11. Kubatko, E.J., Westerink, J.J., Dawson, C.: hp discontinuous Galerkin methods for advection dominated problems in shallow water flow. Comp. Meth. Appl. Mech. Eng. 196, 437–451 (2006)
12. Kubatko, E.J., Bunya, S., Dawson, C., Westerink, J.J., Mirabito, C.: A performance comparison of continuous and discontinuous finite element shallow water models. J. Sci. Comp. (in press)
13. Kubatko, E.J., Bunya, S., Dawson, C., Westerink, J.J.: Dynamic p-adaptive Runge-Kutta discontinuous Galerkin methods for the shallow water equations. Comput. Meth. Appl. Mech. Engrg. (in press)
14. Toro, E.T.: Shock-Capturing Methods for Free-Surface Shallow Flows. John Wiley, Chichester (2001)