AN HPC FRAMEWORK FOR LARGE SCALE SIMULATIONS AND VISUALIZATIONS OF OIL SPILL TRAJECTORIES

Jian Tao[1], Werner Benger[1], Kelin Hu[2], Edwin Mathews[3], Marcel Ritter[4], Peter Diener[1], Carola Kaiser[5], Haihong Zhao[2], Gabrielle Allen[13] and Qin Chen[12]

ABSTRACT

The objective of this work is to build a framework for simulating, analyzing and visualizing oil spill trajectories driven by winds and currents using high performance computing. We adopt a particle model for oil and track the trajectories of oil particles using 2D surface current and wind, which can either be measured directly or estimated with sophisticated coastal storm and ocean circulation models. Our work is built upon the Cactus computational framework. The numerical implementation of the particle model as well as the model coupling modules will become crucial parts of our upcoming full 3D oil spill modeling toolkit. Employing high performance computing and networking, the simulation time can be greatly reduced. Given timely injection of the measurement data, our work can be helpful to predict oil trajectories and facilitate oil clean up, especially after a tropical cyclone.

Keywords: Coastal Hazard; Oil Spill; HPC; Cyberinfrastructure;

INTRODUCTION

Numerical modeling of oil spills is an important tool for tracking the fate and transport of the oil released into the marine environment. With the aid of real time observations or other sophisticated coastal storm models, such numerical simulations can provide useful information such as the extent and magnitude of the spilled oil, the timeline of oil spreading, etc. for quick response in oil spill events. High performance computing systems enable us to carry out such numerical simulations in a more timely and accurate manner. To react to oil spill events such as the Deepwater Horizon catastrophe, being timely in carrying out such numerical simulations is very important. However, the great amounts of observation and simulation data as well as the theoretical and numerical complexity involved in modeling oil spills using high performance computing provide a challenge to the computational science community. Furthermore, numerical modeling for oil spills involves multiple spatial scales, and associated temporal scales, from as small as oil wells to as large as the whole Gulf of Mexico. Different spatial scales have to be considered in order to build a comprehensive 3D oil spill model that can be used to solve realistic problems involved in oil spills. Under the support from the Louisiana Optical Network Initiative under authority of the Louisiana Board of Regents, we carry out this demonstration research and development project to lay out the foundations for an upcoming comprehensive 3D oil spill model. We model and visualize the trajectories of oil spill in severe storms by numerical simulation with high performance computing. The modular design of Cactus enables us to integrate the oil spill model with coastal storm models to carry out numerical simulations of oil spills in different weather conditions.

COMPUTATIONAL INFRASTRUCTURE

[1]Corresponding author: Center for Computation & Technology, Louisiana State University, email: jtao@cct.lsu.edu, fax: (225)578-5362

[2]Civil & Environmental Engineering Department, Louisiana State University

[4]Unit of Hydraulic Engineering, Department of Infrastructure, University of Innsbruck

[5]School of the Coast and Environment, Louisiana State University

[3]Department of Computer Science, Louisiana State University

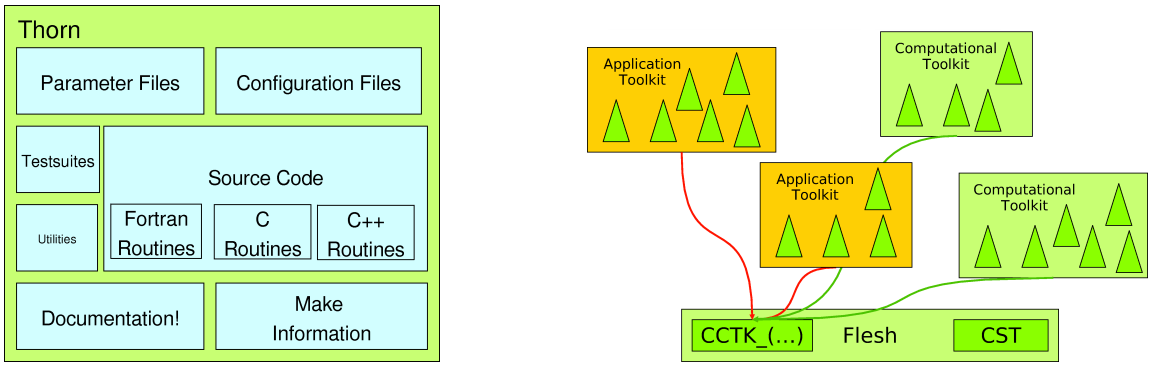

Figure 1: The left diagram shows the internal structure of a typical Cactus thorn. A high level view of a typical Cactus application is shown on the right diagram, where the Cactus Specification Tool (CST) is to provide bindings for the flesh and all Cactus thorns. The Cactus Computational Toolkit (CCTK) provides a range of computational capabilities, such as parallel I/O, data distribution, or checkpointing via the Cactus flesh API.

With the ever increasing complexity in both the hardware and software, the development and maintenance of large scale scientific applications become an intimidating task. Such a task can be easily getting more difficult if one tries to integrate different models with different characteristics. One solution to resolve such application development issues is to build and utilize computational frameworks (or cyberinfrastructures). A computational framework can not only free application developers from lower level programming but also enable effective usage of HPC systems. Programming based on a computational framework is usually more productive because of the abstractions and data structures provided in the framework that are suitable for a particular domain. A successful computational framework often leads to a more collaborative and productive work environment, which is crucial for multidisciplinary research. In this section we will describe the Cactus - Carpet computational framework upon which this work is built.

Cactus Computational Framework

The Cactus Framework [Goodale et al., 2003] was developed to enhance programming productivity and enable large-scale science collaborations. The modular and portable design of Cactus enables scientists and engineers to develop independent modules in Cactus without worrying portability issues on various computing systems. The common infrastructure provide by Cactus also enables the development of scientific codes across different disciplines. This approach emphasizes code reusability, and leads naturally to well constructed interfaces, and well tested and supported software. As the name *Cactus* indicates: the Cactus framework contains a central piece called *flesh*, which provides an infrastructure and interfaces for multiple components or *thorns* in Cactus terminology. Built upon flesh, thorns can provide code for parallelization, mesh refinement, I/O, check-pointing, web servers, coastal modeling, oil spill simulation, etc. The Cactus Computational Toolkit (CCTK) is a collection of thorns that provide basic computational capabilities. The application thorns can make use of the CCTK via calling Cactus flesh API (see Figure 1). In Cactus, the simulation domain is discretized using high order finite differences on block-structured grids. The Carpet library of Cactus enables a basic recursive block-structured AMR algorithm

by Berger-Oliger [Berger and Oliger, 1984]. The time integration schemes used are explicit Runge-Kutta methods and it is provided by the Method of Lines time integrator. The Cactus framework hides the detailed implementation of Carpet and other utility thorns from application developers and separates application development from infrastructure development.

Carpet Adaptive Mesh Refinement Library

The Carpet AMR library [Schnetter et al., 2004, Carpet Website,] is a layer in Cactus to refine parts of the simulation domain in space and/or time, where each refined region is a block-structured regular grid, allowing for efficient internal representations as simple arrays. In addition to mesh refinement, CARPET also provides parallelism and load distribution by distributing grid functions onto processors. To enable parallel execution on multiple processors, our finite differencing stencils require an overlap of several grid points or ghost zones between neighboring processors' sub domains. The inter-process communication is done in Carpet by calling external MPI libraries. In each process, OpenMP is used to further enhance the scalability and performance.

FRAMEWORK FOR MODELING OIL SPILL TRAJECTORIES

The design and development of the oil spill simulation framework follow the same philosophy behind Cactus. We emphasize portability and modularity while improving performance and scalability. We make intensive use of the Cactus computational toolkit for time integration, parallelization, interpolation, I/O, checkpointing, timing, etc.

The oil spill modules can be categorized into two groups: interface modules and application modules. The interface modules define fundamental variables that can be shared among different application modules while the application modules define operations that can be applied to the fundamental variables. While the application modules or mathematical operations can be greatly different depending on models used, the interface or the primary unknows shall stay the same. As shown in Figure 2, we currently define only two interface modules in our framework. Depending on the physical and chemical processes considered, other modules can be added. For simulating the oil spill trajectories on ocean surface, all variables are defined in 2D.

The *CoastalBase* module defines the depth-averaged ocean current velocity and wind velocity 10 meters above ocean surface as fields that depend on the spatial grid at each time step. The variables are initialized by the application module *CoastalInit*, either from detections directly or from data generated in coastal and circulation simulations. In our current setup, we read the mesh file and simulation data from ADCIRC [Luettich and Westerink, 2004, Westerink et al., 2007] and interpolate the data using the inverse distance weighted method from triangular unstructured mesh used in ADCIRC to Cartesian uniform mesh in Cactus. The ocean current velocity and wind velocity can be calculated directly from the fundamental variables defined in other integrated modules. For instance, in building a comprehensive full 3D oil spill model, the 3D velocity field of both occen current and oil in water column shall be calculated during the simulation to estimate the current velocity in order to simulate oil slicks on the surface.

The *OilSpillBase* module defines the positions and advection velocity of oil parcels. Different from the variables defined in CoastalBase, these variables are parcel wise, i.e., they are not treated as Eulerian fields but as properties of each parcel in the Lagrangian point

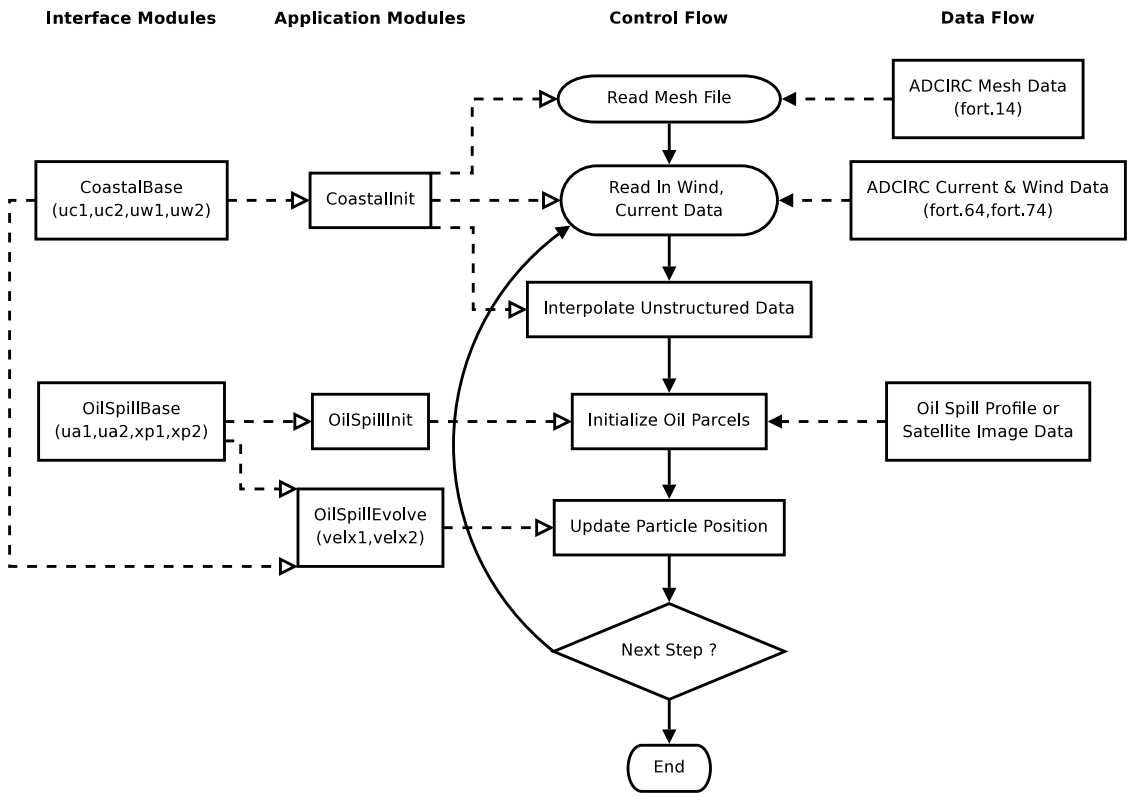

Figure 2: The oil spill modules can be separated into two groups. The interface modules define fundamental variables that can be shared among different modules. The application modules define operations that can be applied to the fundamental variables. Each application module is in charge of one or more tasks in the overall work flow and is responsible for its own input data.

of view. Such a combination of different numerical methods enables us to treat oil spill simulations more efficiently. The *OilSpillInit* module initializes the position and velocity of oil parcels from a given initial profile or some field observation data, which can be processed externally as a spatial distribution of oil.

The evolution of oil parcels is carried out in the *OilSpillEvolve* module. It takes the ocean current velocity and wind velocity from two interface modules respectively after they are updated at each time step by other application modules and update the position of all the oil parcels. For time integration, we use the method of lines provided by the *MoL* module in CCTK. The MoL module provides several time integration schemes, e.g., Rouge Kutta, Iterative Crank Nicholson. These numerical schemes together with other physical and numerical setups can be selected by users in a parameter file. The MoL module provies a mechanism for a certain type of multi-physics coupling where the right hand side of the evolution equations, i.e., the particle velocity in our particle model, can be separated into multiple independent terms which depend on the physical model considered respectively. Each model will just need to update the right hand side without even knowing the existence of other models. Application modules developed upon MoL will be modular by design.

HURRICANE SIMULATION

We improved a parametric analytical wind model for asymmetric hurricanes and merged it with the large-scale background wind field provided by the National Center for Environmental Prediction (NCEP). The improved asymmetric hurricane wind model is developed from the asymmetric Holland-type vortex model [Mattocks and Forbes, 2008]. The model creates a two-dimensional surface wind field based on the National Hurricane Center (NHC) forecast (or observed) hurricane wind point values, namely the maximum wind, radius of maximum wind, the specified (34, 50, and 64-knot) wind intensities and their radii in 4 quadrants. Driven by hurricane wind fields, a fully-coupled wave-surge model (SWAN+ADCIRC) of Dietrich et al is employed to calculate storm surge and depth-integrated currents. The ADCIRC model solves the depth-averaged barotropic shallow-water equation in spherical coordinates using a finite element solution [Luettich and Westerink, 2004, Westerink et al., 2007]. The wave model [Booij et al., 1999] solves the wave action balance equation without any a priori restrictions on the spectrum for the evolution of the wave field. The coupled model can include the interaction of wave and surge in coastal regions. SWAN and ADCIRC use the same unstructured SL15 mesh with about 2.4 M nodes and 4.7M elements. The mesh resolution varies from 24km in the Atlantic Ocean to about 50m in Louisiana and Mississippi. Seven tidal constituents are considered by harmonic constants at the open boundary. The time steps are 1 hr and 1 s for SWAN and ADCIRC, respectively. The coupled model runs in parallel on a supercomputer from the Louisiana Optical Network Initiative (LONI), Queenbee, which has 668 nodes and each node has two 2.33 GHz Quad Core Xeon 64-bit Processors and 8 GB Ram. By using 102 nodes (816 cores), the running time is about 1 hr for the simulation of one actual day.

VISUALIZATION

Proper visualization of the oil spill trajectories addresses two aspects: visual analysis of the simulation data itself and providing a context based on external data. Interfacing external data faces challenges of incompatible data models [Nativi et al., 2004] (systematic obstacles) and file formats [Benger, 2009] (technical obstacles). Based on previous work visualizing hurricane Katrina [Benger et al., 2006] we superimpose the oil spill trajectories on top of satellite imaginary of the Gulf coast. Visual enhancements of the oil transport is provided by generic techniques to visualize vector fields along curves, such as Doppler speckles [Benger et al., 2009a], which allows provides a visual perception of the flow superior to arrow icons. As framework for the exploration of the datasets we employ the Vish visualization shell [Benger et al., 2007], which is very suitable to compute and display path integration lines and evolution fronts within large data sets [Benger et al., 2009b, Bohara et al., 2010b]. While for the particular application here the particle trajectories are only considered within the ocean surface, thus reduced to two dimensions, embedding these data into a three-dimensional environment allows a more realistic interactive impression.

Certain tools for the analysis of pathlines by means of curvature and torsion [Benger and Ritter, 2010] are available in this context, providing indicators for the mixing of fluids [Bohara et al., 2010a], which are oil and ocean water in this case.

NUMERICAL SETUP AND SIMULATION RESULTS

In preparing an oil spill simulation, we take the costal data generated from ADCIRC and SWAN simulation (see section) using the unstructured SL15 mesh with 2.4 M nodes and 4.7M elements and interpolate the depth-averaged current velocity field $\vec{U}_c$ and wind field

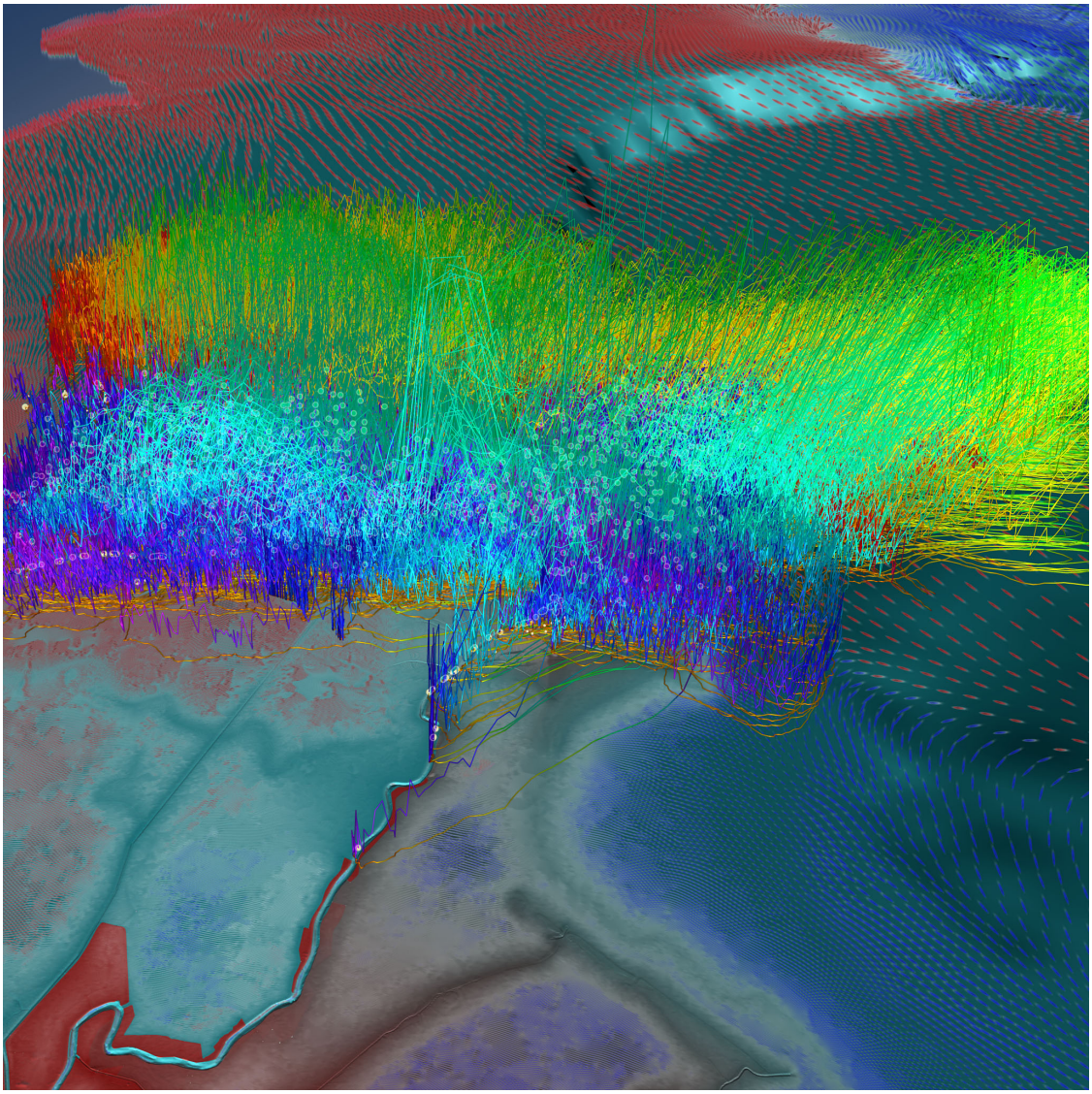

Figure 3: Path-lines of Oil parcels in hurricane Gustav. The trajectories are colored by arc length of the lines using a rainbow color map. The particles move in the XY-plane. An additional scalar field is illustrated by offsetting the line positions in Z-direction, illustrating the distance between neighboring points (ds). End positions of the oil parcels are shown as white points. The ADCIRC model is the source of the triangulated terrain surface and the wind vector-field which is shown using vector-speckles[Benger et al., 2009a] on the terrain grid. The speckles are colored using a color map inspired from the Doppler shift. Red speckles are pointing away and blue speckles are pointing towards the observer.

$\vec{U}_w$ data to a Cartesian uniform grid with a size of 100×100. The inverse distance weighted method is used to carry out the interpolation. We then calculated the advection velocity field $\vec{U}_a = k_c\vec{U}_c + k_w\vec{U}_w$, where k_c and k_w are current and wind drift factor and are set to 1.0 and 0.03 respectively. The initial oil spill profile is created by randomly generating one million parcels near the contaminated area. The advection velocity of each particle is interpolated from the advection velocity field and the postion of the oil parcels is then udpated with the Iterative Crank Nicholson method with a time interval of an hour. For demonstrative purpose, we only consider the advection terms in our simulations.

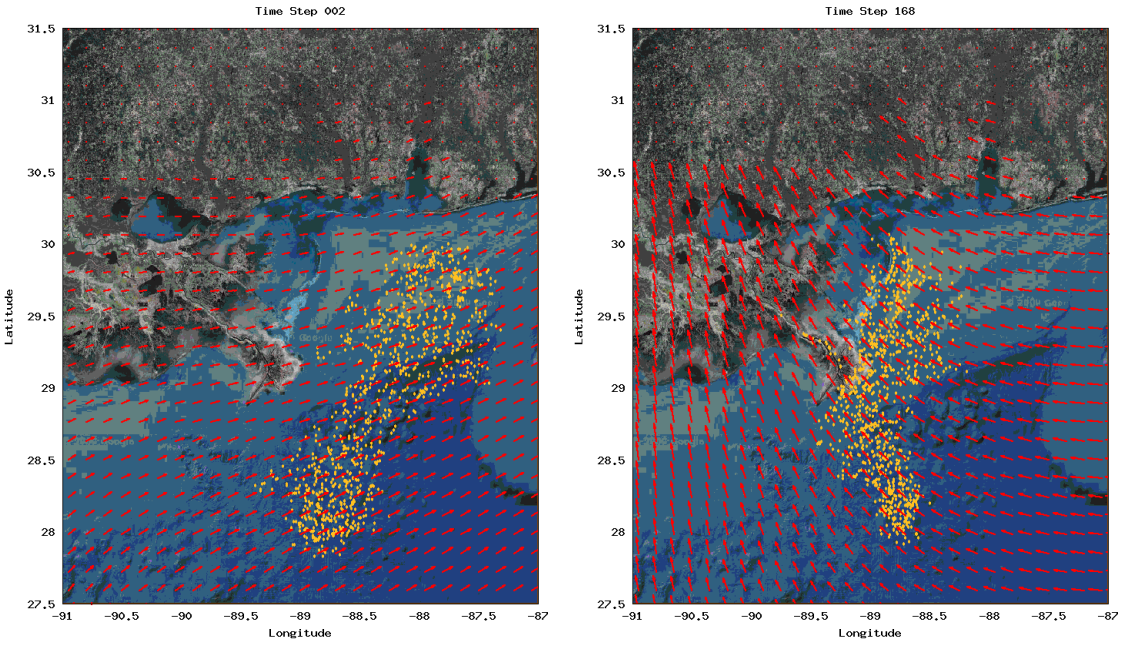

Figure 4: Visualization of a gulf coast oil spill simulation with Gustav hurricane data at two different time steps (downsampled by a factor of 1000). The yellow points represent oil parcels, and the red arrows represent horizontal wind velocity field 10 meters above the ocean surface. The length of the arrows is proportioinal to the wind speed.

CONCLUSION

In this article we present our latest work on building a framework for simulating, analyzing and visualizing oil spill trajectories driven by winds and currents using high performance computing. We take the ocean current velocity and wind data as input and track the trajectories of drifting oil parcels. Based upon the presented framework, we can integrate different coastal and oil spill models for tracking oil spill trajectories. The Cactus-Carpet computational infrastructure used by this work enables us to carry out simulations in parallel on HPC facilities. The presented framework can be used for model coupling and building full 3D oil spill simulations as well.

ACKNOWLEDGMENTS

This work, one of the High Performance Computing (HPC) R&D Demonstration Projects for Oil Spill Disaster Response, is supported by the Louisiana Optical Network Initiative Under Authority

of the Louisiana Board of Regents. The development of the computational cyberinfrastructure is supported by the CyberTools project via NSF awards 701491. This work used the computational resources Eric, Queenbee, Tezpur at LSU/LONI and the NSF TeraGrid under grant number TG-OCE100013. Thanks also go to Soon-Heum Ko, Frank Löffler, and Erik Schnetter for useful discussions.

References

[Benger, 2009] Benger, W. (2009). On safari in the file format djungle - why can't you visualize my data? Computing in Science & Engineering, 11(6):98–102. **Feature Article in "Computing Now"** `http://www.computer.org/portal/web/computingnow/1109/whatsnew/cise`.

[Benger et al., 2007] Benger, W., Ritter, G., and Heinzl, R. (2007). The concepts of vish. In 4^{th} High-End Visualization Workshop, Obergurgl, Tyrol, Austria, June 18-21, 2007, page in print. Berlin, Lehmanns Media-LOB.de.

[Benger et al., 2009a] Benger, W., Ritter, G., Su, S., Nikitopoulos, D. E., Walker, E., Acharya, S., Roy, S., Harhad, F., and Kapferer, W. (2009a). Doppler speckles - a multi-purpose vectorfield visualization technique for arbitrary meshes. In CGVR'09 - The 2009 International Conference on Computer Graphics and Virtual Reality.

[Benger and Ritter, 2010] Benger, W. and Ritter, M. (2010). Using Geometric Algebra for Visualizing Integral Curves. In Hitzer, E. M. and Skala, V., editors, GraVisMa 2010 - Computer Graphics, Vision and Mathematics for Scientific Computing. Union Agency - Science Press.

[Benger et al., 2009b] Benger, W., Ritter, M., Acharya, S., Roy, S., and Jijao, F. (2009b). Fiberbundle-based visualization of a stir tank fluid. In 17^{th} International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pages 117–124.

[Benger et al., 2006] Benger, W., Venkataraman, S., Long, A., Allen, G., Beck, S. D., Brodowicz, M., MacLaren, J., and Seidel, E. (2006). Visualizing Katrina - Merging Computer Simulations with Observations. In Workshop on state-of-the-art in scientific and parallel computing, Umeå, Sweden, June 18-21, 2006, pages 340–350. Lecture Notes in Computer Science (LNCS), Springer Verlag.

[Berger and Oliger, 1984] Berger, M. J. and Oliger, J. (1984). Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys., 53:484–512.

[Bohara et al., 2010a] Bohara, B., Benger, W., Ritter, M., Roy, S., Brener, N., and Acharya, S. (2010a). Time-curvature and time-torsion of virtual bubbles as fluid mixing indicators. IADIS Computer Graphics, Visualization, Computer Vision and Image Processing 2010 (CGVCVIP 2010).

[Bohara et al., 2010b] Bohara, B., Harhad, F., Benger, W., Brener, N., Iyengar, S., Ritter, M., Liu, K., Ullmer, B., Shetty, N., Natesan, V., Cruz-Neira, C., Acharya, S., and Roy, S. (2010b). Evolving time surfaces in a virtual stirred tank. Journal of WSCG, 18(1-3):121–128.

[Booij et al., 1999] Booij, N., Ris, R. C., and Holthuijsen, L. H. (1999). A third-generation wave model for coastal regions, part 1, model description and validation. Journal of Geophysical Research, 104 (C4):7649–7666.

[Carpet Website,] Carpet Website. Adaptive mesh refinement with Carpet. `http://www.carpetcode.org/`.

[Goodale et al., 2003] Goodale, T., Allen, G., Lanfermann, G., Massó, J., Radke, T., Seidel, E., and Shalf, J. (2003). The Cactus framework and toolkit: Design and applications. In High Performance Computing for Computational Science - VECPAR 2002, 5th International Conference, Porto, Portugal, June 26-28, 2002, pages 197–227, Berlin. Springer.

[Hutanu et al., 2010] Hutanu, A., Schnetter, E., Benger, W., Bentivegna, E., Clary, A., Diener, P., Ge, J., Kooima, R., Korobkin, O., Liu, K., Loffler, F., Paruchuri, R., Tao, J., Toole, C., Yates, A., and Allen, G. (2010). Large Scale Problem Solving Using Automatic Code Generation and Distributed Visualization. Scientific International Journal for Parallel and Distributed Computing, 11 (2):124016.

[Luettich and Westerink, 2004] Luettich, R. A. and Westerink, J. (2004). Formulation and numerical implementation of the 2d/3d adcirc finite element model version 44.xx.

[Mattocks and Forbes, 2008] Mattocks, C. and Forbes, C. (2008). A real-time, event-triggered storm surge forecasting system for the state of north carolina. Ocean Modelling, 25:95–119.

[Nativi et al., 2004] Nativi, S., Blumenthal, B., Habermann, T., Hertzmann, D., Raskin, R., Caron, J., Domenico, B., Ho, Y., and Weber, J. (2004). Differences among the data models used by the geographic information systems and atmospheric science communities. In Proceedings American Meteorological Society - 20th Interactive Image Processing Systems Conference.

[Schnetter et al., 2004] Schnetter, E., Hawley, S. H., and Hawke, I. (2004). Evolutions in 3D numerical relativity using fixed mesh refinement. Class. Quantum Grav., 21(6):1465–1488. gr-qc/0310042.

[Westerink et al., 2007] Westerink, J., Luettich, R., Feyen, J., Atikinson, J., C. Dawson, H. R., Powell, M., Dunion, J., Kubatko, E., and Pourtaheri, H. (2007). A basin to channel scale unstructured grid hurricane storm surge model applied to southern louisiana. Monthly Weather Review, (136):833–864.