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

Jian Tao[1,*], Werner Benger[1], Kelin Hu[2], Edwin Mathews[1,3], Marcel Ritter[1,4], Peter Diener[1,5], Carola Kaiser[1,6] , Haihong Zhao[2], Gabrielle Allen[1,7] and Qin Chen[1,2]

ABSTRACT

The objective of this work is to build a high performance computing framework for simulating, analyzing and visualizing oil spill trajectories driven by winds and ocean currents. We adopt a particle model for oil and track the trajectories of oil particles using 2D surface currents and winds, 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; Cactus; Cyberinfrastructure

INTRODUCTION

The accurate numerical modeling of oil spills is an important capability for tracking the fate and transport of oil released into a marine environment. With the integration of data from real time observations or sophisticated coastal storm models, such numerical simulations can provide information about the extent and magnitude of the spilled oil, the timeline of oil spreading, etc. for quick response to 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 configuring and carrying out such numerical simulations is very important. However, the large amounts of observational 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 and temporal scales requiring resolution that stretches from oil wells to the whole of the Gulf of Mexico. Different spatial scales have to be considered in order to build a comprehensive 3D oil spill model that can be deployed to solve real world problems. With support from the Louisiana

[1] Center for Computation & Technology, Louisiana State University,
[*] Corresponding author, email: jtao@cct.lsu.edu, fax: (225)578-5362
[2] Department of Civil & Environmental Engineering, Louisiana State University
[3] Department of Mechanical Engineering, Louisiana State University
[4] Unit of Hydraulic Engineering, Department of Infrastructure, University of Innsbruck
[5] Department of Physics, Louisiana State University
[6] School of the Coast and Environment, Louisiana State University
[7] Department of Computer Science, Louisiana State University

Optical Network Initiative under authority of the Louisiana Board of Regents, we have carried out a demonstration research and development project that lays the foundations for a planned comprehensive 3D oil spill model. Here, we model and visualize trajectories of oil spills in severe storms using numerical simulation and high performance computing. The modular design of our software, that uses the Cactus framework, enables us to easily integrate the oil spill model with coastal storm models to carry out highly scalable numerical simulations of oil spills in different weather conditions.

COMPUTATIONAL INFRASTRUCTURE

With the increasing complexity of both hardware and software, the development and maintenance of large scale scientific applications is currently an intimidating task. This task becomes even more complex when we need to integrate together different physics models each with their own varying characteristics. One solution to enable such application development issues is to build on computational frameworks, which can free application developers from low-level programming, increase code re-use and enable effective usage of HPC systems. Programming based on a computational framework can be more productive due to the abstractions and data structures provided by the framework that are suitable for a particular domain. A successful computational framework also leads to a more collaborative and productive work environment, which is crucial for multidisciplinary research. In this section we will describe the Cactus computational framework upon which this work is built.

CACTUS COMPUTATIONAL FRAMEWORK

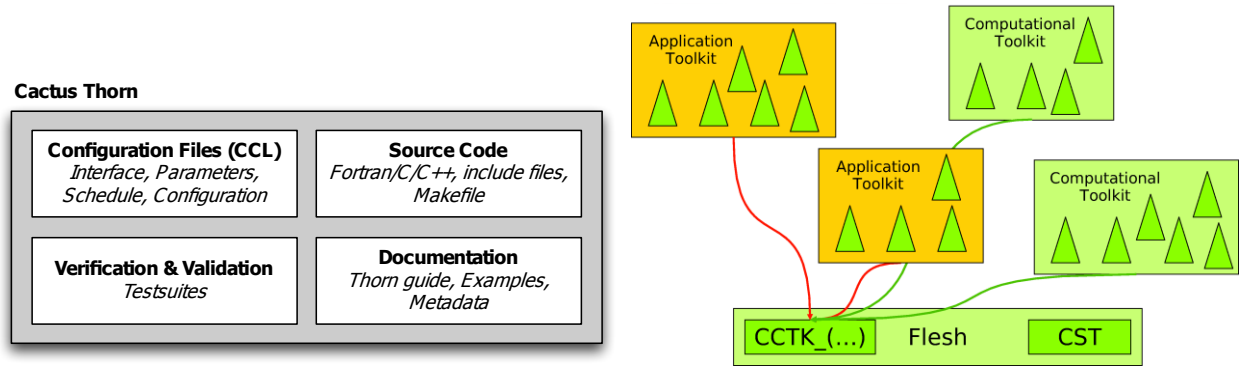

Figure 1: [Left] Internal structure of a typical Cactus component (thorn). [Right] High level view of a typical Cactus application, where the Cactus Specification Tool (CST) provides bindings between thorns and the flesh. The Cactus Computational Toolkit (CCTK) provides a range of computational capabilities, such as parallel I/O, data distribution, or checkpointing via the Cactus flesh API.

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 about portability issues on different computing systems. The common infrastructure

provided by Cactus also enables the development of scientific codes that reach across different disciplines. This approach emphasizes code reusability, leads naturally to well designed interfaces, and to well tested and supported software. As the name *Cactus* indicates: the Cactus framework contains a central part called the *flesh*, which provides an infrastructure and interfaces to multiple components or *thorns* in Cactus terminology. Built upon the flesh, thorns can provide capabilities 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 make use of the CCTK via abstract interfaces such as the flesh API (see Figure 1). In Cactus, the simulation domain can be discretized using high order finite differences on block-structured grids. The Carpet library for Cactus provides a parallel implementation of 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 are 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.

VISUALIZATION INFRASTRUCTURE

For three-dimensional visualization we employ the Vish Visualization Shell, a highly modular research framework to implement visualization algorithms. Similarly to Cactus, Vish provides a micro-kernel with plugins which are loaded at runtime, allowing developers to independently implement specific aspects without interfering each other. As a framework it is designed for exploratory scientific visualization rather than providing static solutions for a limited set of data. We apply experimental visualization methods that had been developed for other application areas to find features and properties in this oil spill simulation data set that are not obvious through conventional visualization approaches. As Vish allows overriding each aspect of the visualization on a very fine level including hardware-oriented GPU programming, we achieve high performance and flexibility. For instance as part of this exploration we experimented with using a scalar field along the particle trajectories as height, similar to a height field, in order to display particle properties better than just colorization. The method of "Doppler speckles", originally developed to be applied upon astrophysical datasets, turns out to be useful finely resolved vector fields where vector arrows are of limited use due to increasing visual clutter. Integration of data sets from various sources is addressed via converting them into HDF5 using the F5 layout, which allows efficient handling of massive datasets through one common interface.

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 unknowns 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.

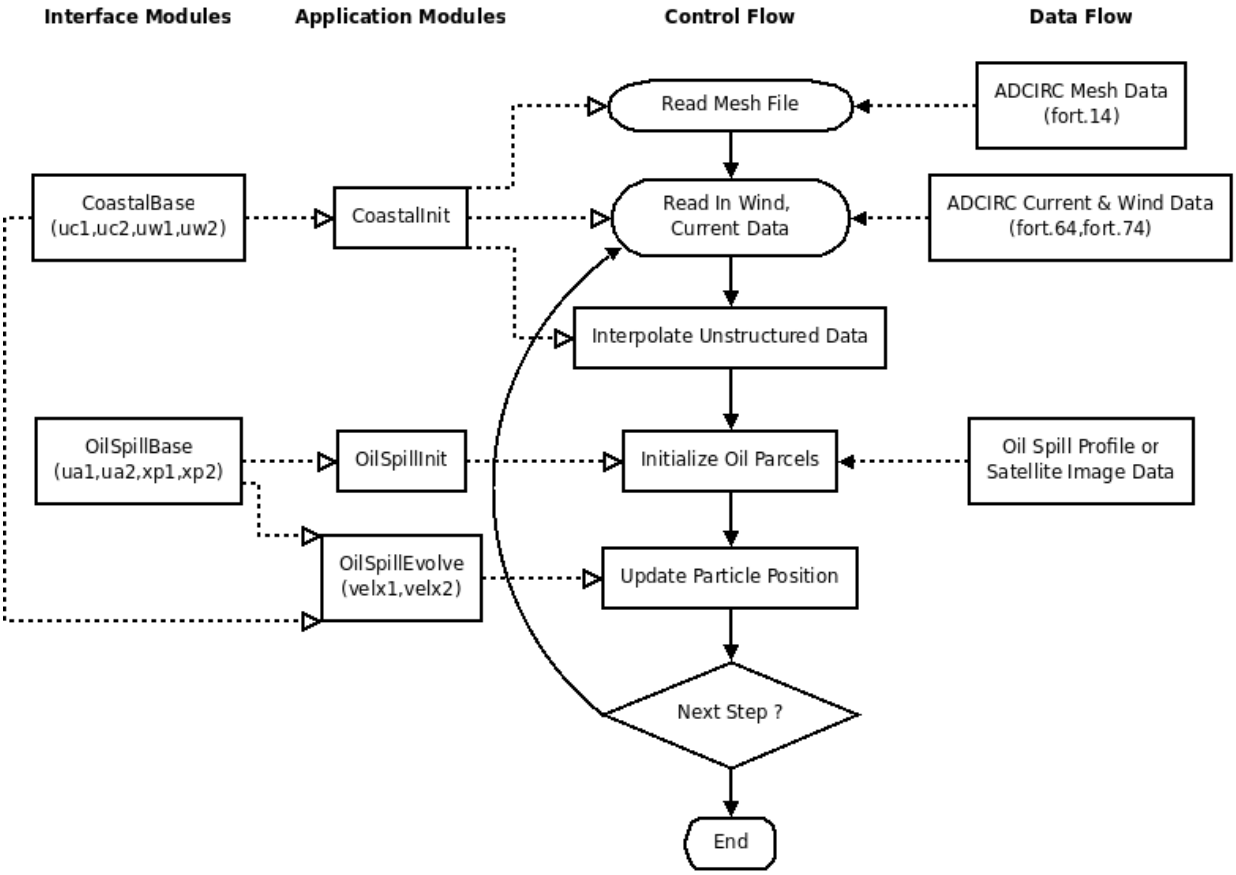

Figure 2: The oil spill modules can be separated into two groups. The interface modules define fundamental variables that can be shared among different modules. The application modules define operations that can be applied to the fundamental variables. Each application module is in charge of one or more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

The *CoastalBase* module defines the depth-averaged ocean current velocity and wind velocity 10 meters above the ocean surface as fields that depend on the spatial grid at each time

step. The application module *CoastalInit* initialize these variables, either from direct observations or from data generated in coastal and circulation simulations. In our current setup, we read in the mesh file and simulation data from ADCIRC (Luettich and Westerink, 2004; Westerink et al., 2008) and interpolate this data using the inverse distance weighted method from triangular unstructured mesh used in ADCIRC to the 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 ocean 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. Differently to 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 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., Ronge Kutta, Iterative Crank Nicholson. Users can select these numerical schemes together with other physical and numerical setups through the parameter file. The MoL module provides 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*. (2010) 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., 2008). 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 the Queen Bee supercomputer provided by the Louisiana Optical Network Initiative (LONI). Queen Bee has 668 nodes with each node containing two 2.33 GHz Quad Core Xeon 64-bit Processors and 8 GB Ram. Using 102 nodes (816 cores), the wallclock time is about 1 hr for the simulation of one actual day.

Figure 3 shows a snapshot of storm surge distribution during Hurricane Gustav. At this time (10:00 UTC, 09/01/2008), the center of the hurricane was near the Louisiana coast. The eastern winds to the front right of the hurricane caused a surge setup (about 3m) at the Breton Sound and the east bank of Mississippi River. The northern and north-eastern winds to the front left of the hurricane blew the water offshore and caused about 1m setdown of storm surge along the Louisiana coast (from 92^0 W to 90.5^0 W).

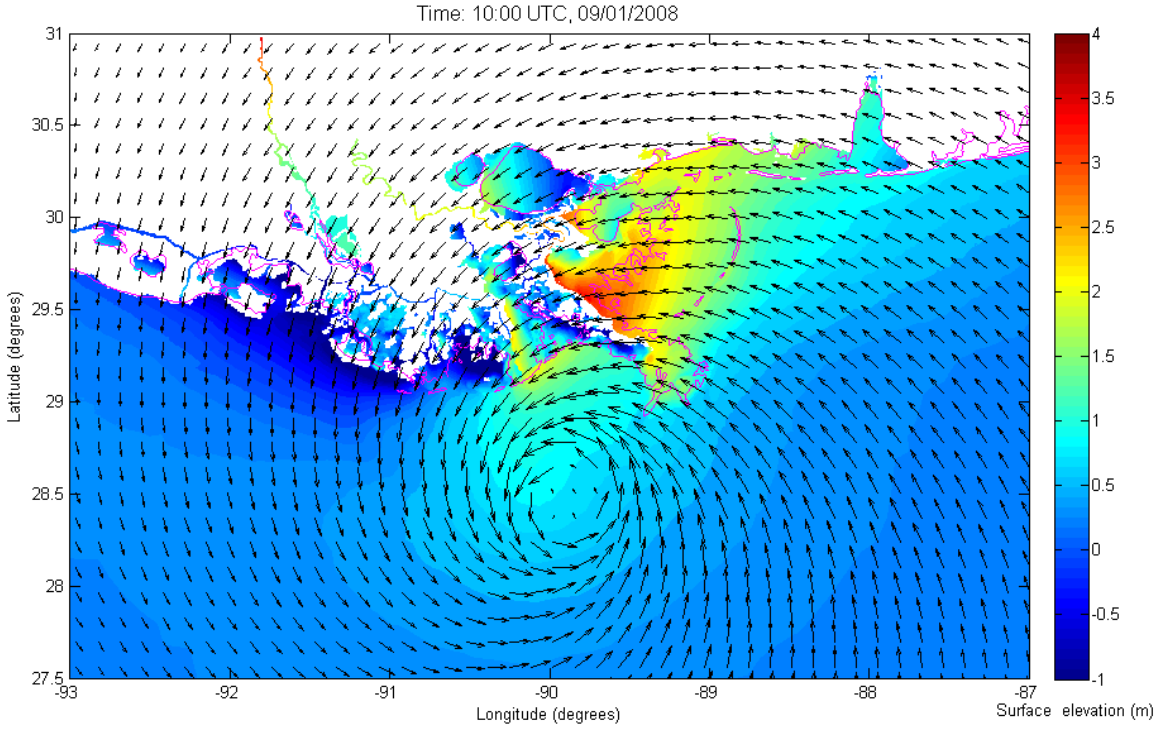

Figure 3: A snapshot of storm surge distribution near Louisiana coast at the time of 10:00 UTC, 09/01/2008, during Hurricane Gustav. The interval of contour line is 0.1m. The black arrows denote the wind vectors at the same time.

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 imagery 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 provides a visual perception of the flow that is superior to arrow icons. The Vish visualization

shell (Benger *et al*., 2007) is used as a framework for visualization, which is very suitable for computing and displaying 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 reducing the problems to two dimensions, embedding these data into a three-dimensional environment allows a more realistic interactive impression.

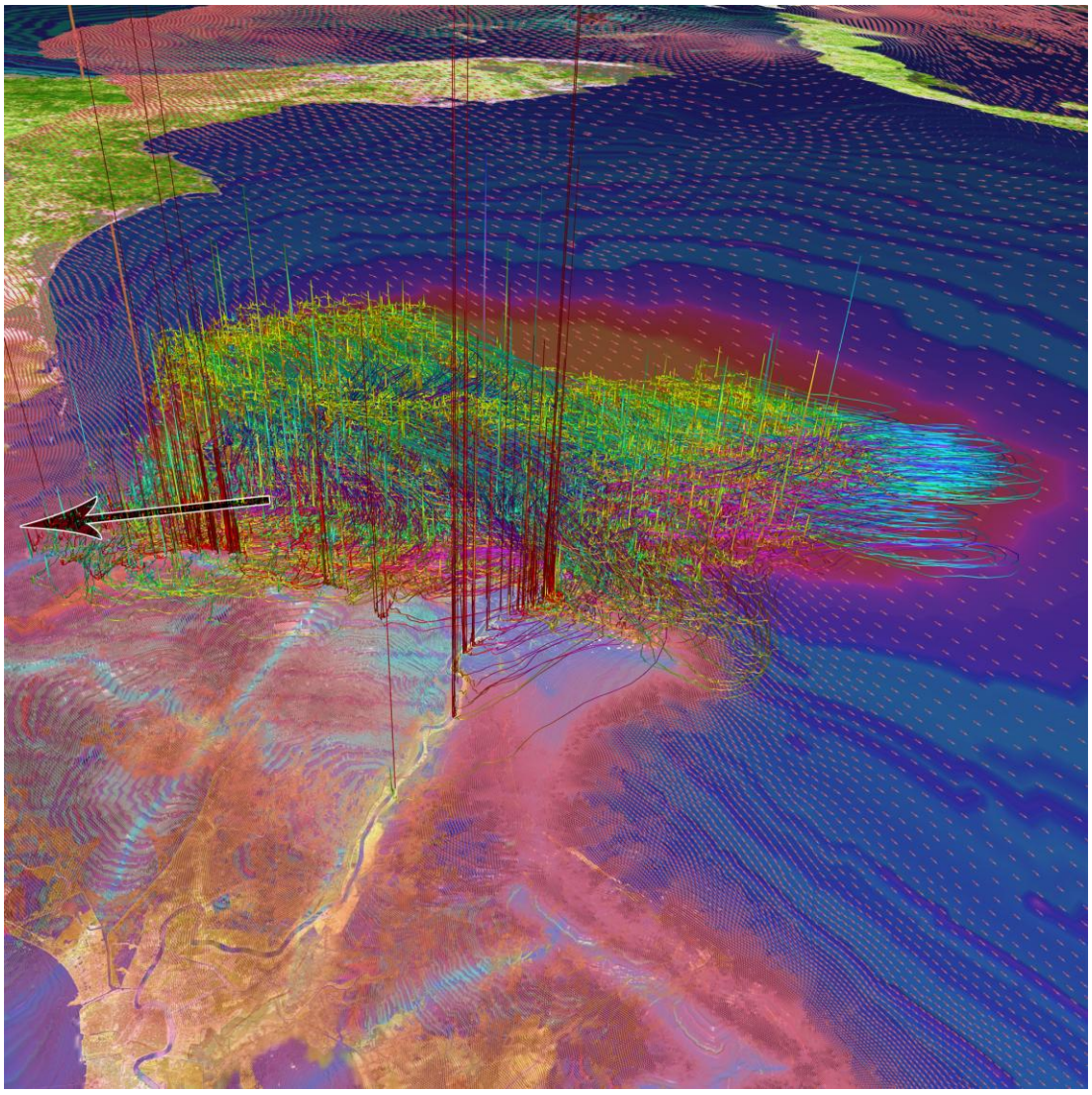

Figure 4: Path-lines of Oil parcels in hurricane Gustav simulated in Cactus and viusualized in Vish. The path-lines are colored by arclength of the lines. The particles move in the XY-plane. An additional scalar field is illustrated by offsetting the line positions in Z-direction, illustrating the curvature of the trajectories. This marks positions of the particles with high changes in directions. The ADCIRC model is the source of the elevated water surface that is shown as an elevated and color-mapped surface. Also the wind vector-field which is shown using vector-speckles[Benger et al., 2009a] on the terrain grid is provided by the ADCIRC data. An aligned 50m resolution satellite image shows the Mississippi delta in the foreground. A 500m resolution covers the background. The arrow illustrates North direction.

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

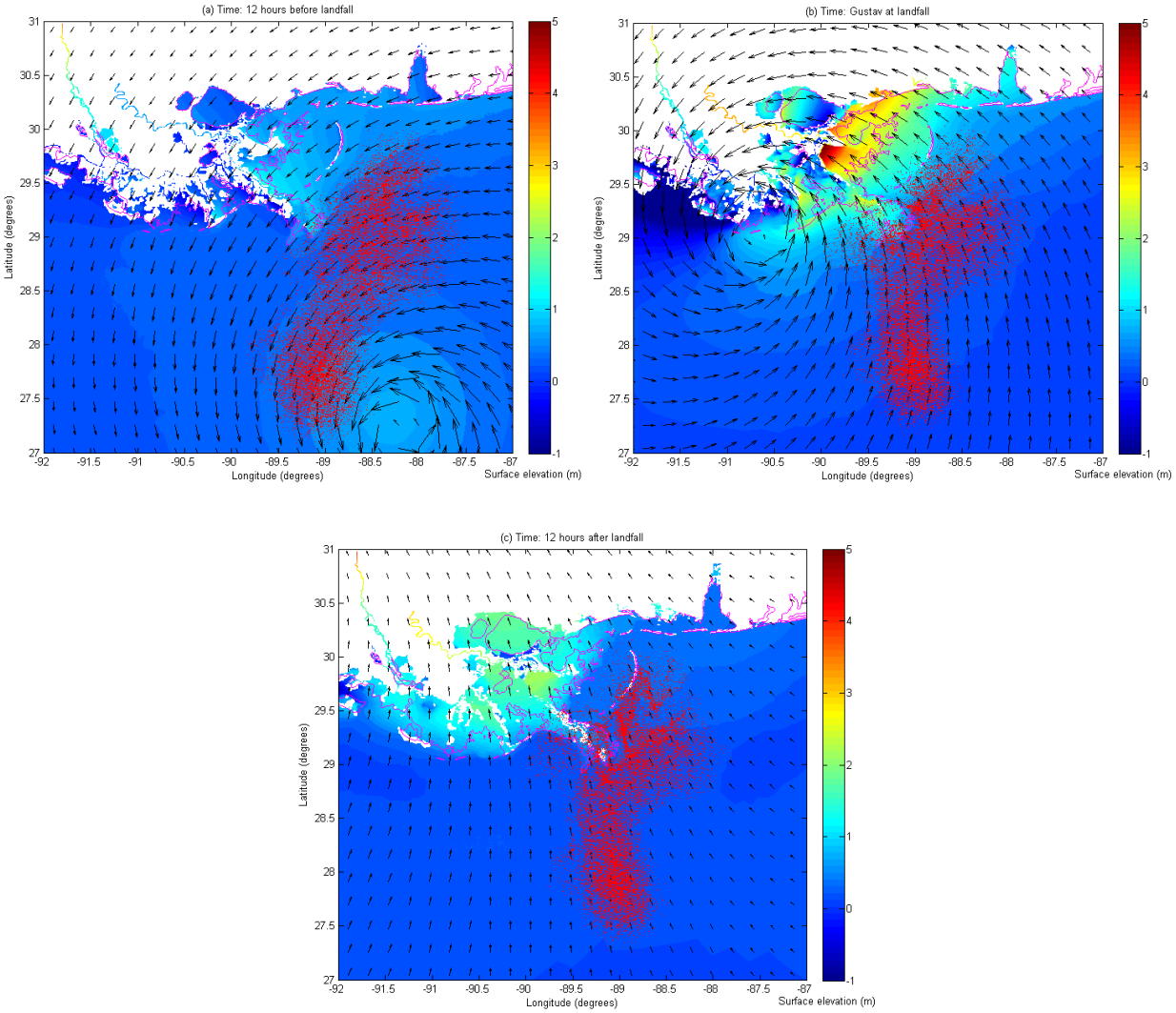

Figure 5: Visualization of a gulf coast oil spill simulation with Gustav hurricane data at three different time steps (down-sampled by a factor of 50). The red points represent oil parcels, and the black arrows represent horizontal wind velocity field 10 meters above the ocean surface. The length of the arrows is proportional to the wind speed. The background is the storm surge distribution. The interval of contour line is 0.1m.

In preparing an oil spill simulation, we took the Hurricane Gustav data from ADCIRC and SWAN simulations (see section 'Hurricane Simulation') using the unstructured SL15 mesh with 2.4M nodes and 4.7M elements. We then interpolated the depth-averaged current velocity field

$\vec{U}_C$ and wind field $\vec{U}_W$ data onto a 100×100 Cartesian uniform grid. The inverse distance weighted method is used to carry out the interpolation. We 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 the current and wind drift factor and were set to 1.0 and 0.03 respectively. The initial oil spill profile was created by randomly generating 1,000,000 oil parcels near the contaminated area. The advection velocity of each oil particle was interpolated from the advection velocity field and the position of the oil parcels was then updated using the Iterative Crank Nicholson method with a time interval of an hour. Only the advection terms were considered in our simulations. We carried out a demonstrative run in parallel with 4 MPI processes on a workstation with two dual core AMD Opteron processors and 8 GB memory. On the workstation, each time step took about 40 seconds after the weight function for interpolation was calculated and stored in memory before the time integration starts. The calculation of the weight function alone took about 20 minutes. The simulation results are shown in Figure 5. At the time of 12 hours before the landfall, the oil parcels moved southward due to the counter-clockwise hurricane winds at the northwest to the hurricane center. When Gustav made landfall, the parcels moved toward shoreline under the southern and south-eastern winds. At the time of 12 hours after the landfall, although the barrier islands blocked most of the parcels, some parcels still can move into the Breton Sound and its adjacent water.

CONCLUSION

In this article we have presented our recent work towards building a framework for simulating, analyzing and visualizing oil spill trajectories driven by winds and ocean currents using high performance computing. We took the ocean current velocity and wind data as input and tracked 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 oil spill simulations in parallel. It also gets us ready to address multiple scale problems in building a planned comprehensive 3D oil spill model with an adaptive mesh refinement library fully integrated.

ACKNOWLEDGMENTS

This work, a High Performance Computing (HPC) R&D Demonstration Project 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 award 701491. This work used the computational resources Eric, Queenbee, Tezpur at LSU/LONI and the NSF TeraGrid under grant number TGOCE100013. Thanks also go to Soon-Heum Ko, Frank Loeffler, and Erik Schnetter for useful discussions. The study has been supported in part by a grant from the Office of Naval Research Coastal Geosciences Program (N00014-07-1-0955).

REFERENCES

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 Articlein “Computing Now” http://www.computer.org/portal/web/computingnow/1109/whatsnew/cise.

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

Benger, W., G. Ritter, S. Su, D.E. Nikitopoulos, E. Walker, S. Acharya, S. Roy, F. Harhad, and W. Kapferer (2009a), “Doppler speckles - a multipurpose vectorfield visualization technique for arbitrary meshes”, In CGVR’09 - The 2009 International Conference on Computer Graphics and Virtual Reality.

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

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

Benger, W., S. Venkataraman, A. Long, G. Allen, S.D. Beck, M. Brodowicz, J. MacLaren, and E. Seidel (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, M. J. and J. Oliger (1984), “Adaptive mesh refinement for hyperbolic partial differential equations”, Journal of Computational Physics. **53**, 484–512.

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

Booij, N., R.C. Ris, and L.H. Holthuijsen (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, “Adaptive mesh refinement with Carpet”, http://www.carpetcode.org/.

Goodale, T., G. Allen, G. Lanfermann, J. Massó, T, Radke, E. Seidel, and J. Shalf (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.

Dietrich, J. C., S. Bunya,J. J. Westrink, B. A. Ebersole, J. M. Smith, J. H. Atkinson, R. Jensen, D. T. Resio, R. A. Luetich, C. Dawson, V. J. Cardone, A. T. Cox, M. D. Powell, H. J. Westerink, and H. J. Roberts (2010), “A High-Resolution Coupled Riverine Flow, Tide, Wind, Wind Wave, and Storm Surge Model for Southern Louisiana and Mississippi. Part II: Synoptic Description and Analysis of Hurricanes Katrina and Rita”, Monthly Weather Review, **138**, 378-404

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

Luettich, R. A. and J. Westerink (2004), "Formulation and numerical implementation of the 2D/3D ADCIRC finite element model version 44.xx", 74pp. [Available online at http://adcirc.org/adcirc_theory_2004_12_08.pdf.]

Mattocks, C., and C. Forbes (2008), "A real-time, event-triggered storm surge forecasting system for the state of North Carolina", Ocean Modelling, **25**, 95-119.

Nativi, S., B. Blumenthal, T. Habermann, D. Hertzmann, R. Raskin, J. Caron, B. Domenico, Y. Ho, and J. Weber (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, E., S.H. Hawley, and I. Hawke (2004), "Evolutions in 3D numerical relativity using fixed mesh refinement", Class. Quantum Grav., **21**(6), 1465–1488. gr-qc/0310042.

Westerink, J., R. Luettich, J. Feyen, J. Atikinson, C. Dawson, H. Roberts, M. Powell, J. Dunion, E. Kubatko, and H. Pourtaheri (2008), "A Basin to Channel Scale Unstructured Grid Hurricane Storm Surge Model Applied to Southern Louisiana", Monthly Weather Review, **136**, 833-864.