Journal of Coastal Research	SI 50	pg - pg	ICS2007 (Proceedings)	Australia	ISSN

Towards a faster and improved ADCIRC (ADvanced Multi-Dimensional CIRCulation) model

E. Ceyhan‡, P. Basuchowdhuri†, T. Judeh†, S. Ou‡, B. Estrade‡, and T. Kosar†‡

†Department of Computer Science,
Louisiana State University,
Baton Rouge, 70803,USA
E-mail: (pbasuc1,tjudeh1)@lsu.edu

‡ Center for Computation and Technology(CCT),
Louisiana State University
Baton Rouge, 70803,USA
E-mail: (eceyhan,estrabd,kosar)@cct.lsu.edu
sou1@lsu.edu

ABSTRACT

Given the aftermaths of Hurricane Katrina in New Orleans and the surrounding parishes, the need to predict tropical storms and hurricanes has increased multifold. Accurate predictions may assist decision makers to implement appropriate evacuation plans necessary for the wellbeing of citizens. Coastal modeling is a key tool used to foresee hurricanes, storm surge, and flooding. Of the many models implemented, ADCIRC, a project of the University of North Carolina, seems to hold much promise. Our objectives in this study were multifold. First, we ascertained the current limitations of the ADCIRC model. We achieved this goal through a variety of means including ADCIRC benchmarks on single and multiple clusters as well as a gprof profile of ADCIRC on a single cluster. Then, based on the information obtained from these efforts, we suggested appropriate improvements to optimize ADCIRC. With the achievement of these objectives, we hope ADCIRC will become a vital tool in developing accurate evacuation plans under duress that will aid in saving a countless number of lives.

INDEX WORDS: *ADCIRC, Coastal Modeling, Tightly Coupled Applications*

INTRODUCTION

In areas where hurricanes, storm surges, and flooding occur frequently, developing appropriate evacuation plans is a must. Coastal modeling allows for the study of the complex coastal systems. Consequently, scientists can then utilize these models as aids in predicting hurricanes, storm surges, and flooding. A number of models have been implemented in this spirit [LYNCH et al., 1996, LUETTICH et al., 1992]. Following these models, it may very well be possible to take preventive measures to minimize the loss in event of a hurricane, storm surge, and flooding.

Developed at the University of North Carolina, ADCIRC (ADvanced Multi-Dimensional CIRCulation Model for Shelves, Coasts and Estuaries) is a multi-dimensional, depth integrated, barotropic time-dependent long wave, finite element-based hydrodynamic circulation code. It can be run either as a two-dimensional depth integrated (2DDI) model or as a three-dimensional (3D) model. ADCIRC-2DDI solves the vertically-integrated momentum equations to determine the depth-averaged velocity. 3D versions of ADCIRC solve the velocity components in the coordinate direction, x, y, and z over generalized stretched vertical coordinate system by utilizing the shallow water form of the moment equation. In either case, elevation is attained from the solution of the depth-integrated continuity equation in GWCE form. Velocity is attained from the solution of either momentum equations. All nonlinear terms have been kept in these equations.

The purpose of ADCIRC is to solve time dependent, free surface circulation and transport problems in two and three dimensions. These problems are handled using computer programs that make use of the finite element method in space while allowing the use of highly flexible unstructured grids [LUETTICH et al., 1992].

According to [LUETTICH et al., 1992], ADCIRC application consists of :

- modeling tides and wind driven circulation
- analysis of hurricane storm surge and flooding
- dredging feasibility and material disposal studies
- larval transport studies
- near shore marine operations

The ADCIRC model has included several applications in significant areas. Currently, ADCIRC can only be executed on a tightly coupled cluster. Our major goal in this paper is ascertain the current limitations of this model, and then suggest possible improvements and optimize ADCIRC.

The outline of the article is as follows. First, we provide details on related work. Then, we briefly introduce the ADCIRC model theory. We then touch upon advantages and disadvantages of this model. Afterwards, we present the ADCIRC benchmark results on a single high-performance compute cluster, the gprof profile for ADCIRC on that cluster, and initial benchmark results for ADCIRC on multiple clusters. Finally, we conclude with a discussion of current system status and future plans.

RELATED WORK

There are a variety of fields of work related to ADCIRC. For our purposes, though, we examine the literature on decoupling tightly coupled applications that allow single cluster applications to be run on multiple clusters. Given that multiple clusters exist these days, being able to run ADCIRC on multiple clusters where communication does not overcome computation will allow for a better performance of ADCIRC. One effort in decoupling a tightly coupled application can be found in [BARNARD et al., 1999]. In this case the authors implemented a computational fluid dynamic

(CFD) algorithm, OVERFLOW, into a distributive algorithm to be executed on the Information Power Grid. OVERFLOW is heavily used by NASA, and a time is coming where the resources of a single supercomputer would not suffice for the purposes of NASA. To achieve this goal of executing OVERFLOW on multiple clusters, significant work was performed. First, the authors had a framework in which the distributed algorithm can be applied via Globus and the Information Power Grid. More interesting for our purposes, though, is the manner in which the tightly coupled iterative solver was slightly modified to allow for the implementation of OVERFLOW in a multiple cluster environment.

This modified algorithm's property can be summed up in three major points. First, one needs to briefly illustrate traditional parallel flow solvers. These traditional solvers synchronize and communicate boundary values for a particular timestep N. After the synchronization, the values are used to compute the boundary values for timestep N + 1 using the current values of N.

In the modified approach, nodes asynchronously communicate the boundary values for a timestep N. In the meantime, the N + 1 timestep is computed using the N – 1 timestep boundary values. This procedure allows for hiding the communication latency associated with processes running on multiple clusters. However, a drawback is that convergence to a solution takes much longer to occur. Another approach found in [MECHOSO et al., 1999] involves modularizing the tightly coupled application into different components that may then be run independently of each other. In [MECHOSO et al., 1999], the tightly coupled application was split into three components. Two of the components relied on data provided by the third and did not need to interact with one another. As a result, these two components can be implemented in parallel to achieve a significant speedup.

Also, in [MESRI et al, 2005] work has been done on a new graph/mesh partitioner called MeshMigration. This work would no doubt prove valuable to applications such as ADCIRC and other mesh based applications. Whereas applications such as Metis, Chaco, and Jostle have long existed and been used for portioning meshes, these applications make a very limiting assumption that all resources found in the computing environment are uniform. This limitation makes adapting a Metis-heavy application, such as ADCIRC, to a multicluster/grid environment challenging.

MeshMigration was developed to work in a heterogeneous environment. However, one must ask how significant were load balancing strategies in obtaining the results seen in [MESRI et al, 2005]. As for now, though, Metis remains the partitioner for ADCIRC as well as many other applications.

ADCIRC MODEL: THEORETICAL ASPECTS

ADCIRC is a highly developed computer program to solve the equations of moving fluid on a rotating earth. The related equations were formulated by using the traditional hydrostatic pressure and Boussinesq approximations and have been discretized in space using the finite element (FE) method and in time using the finite difference (FD) method.

ADCIRC is generally run either as a two dimensional depth integrated (2DDI) model or as a three dimensional (3D) model. It can be run in either Cartesian or spherical coordinate systems, respectively. Elevation (needed in both cases) is obtained by solving the depth-integrated continuity equation in GWCE (Generalized Wave-Continuity Equation) form. Velocity is obtained by solving either the 2DDI or 3D momentum equations.

The GWCE is solved by using a consistent or a lumped mass matrix (via a compiler flag) and an implicit or explicit time stepping scheme (via variable time weighting coefficients). In case of a lumped, fully explicit formulation no matrix solver is necessary. For all other cases, the GWCE is solved by using the Jacobi preconditioned iterative solver from the ITPACKV 2D package. The 2DDI momentum equations require no matrix solver as they are lumped. In 3D, vertical diffusion is treated implicitly and the vertical mass matrix is not lumped. So, a solution of a complex, tri-diagonal matrix problem is required over the vertical at every horizontal node [LUETTICH et al., 2004].

ADCIRC's algorithmic design criteria have a very low numerical damping model that allows model parameters to be based on physically relevant values. The algorithm should be consistent with the governing equations. It should also be at least second order accurate and also robust in nature.

Furthermore, ADCIRC has been optimized by unrolling loops for enhanced performance on multiple computer architectures. ADCIRC includes MPI library calls to allow it to operate at high efficiency (typically more than 90 percent) on parallel computer architectures.

In a single processor or parallel environment (high performance machines), ADCIRC runs efficiently to solve large unstructured grids. ADCIRC's speed up is linear when executed on up to 256 processors for a mid-sized mesh [ADCIRC 2006]. However, when one exceeds 256 processors, the communication overhead starts to dominate over the computational efficiency. For larger meshes, ADCIRC can easily scale up to 1000 processors.

ADCIRC is appropriate for tidal prediction because of its computational efficiency partly because of implementing the finite element method and the use of an iterative sparse matrix solver. Simulations that extend from months to a year can be completed and employ 30-second time-step increments expediently and also without unreasonable storage requirements. For example, a model of the western North Atlantic Ocean containing 36,185 points completed a 250-day simulation of the tides after using only 103 hours of CPU on a CRAY SV1 (single-processor speed of 500 MHz). As such, real-time forecasting (e.g. 48 hour forecasts) is well within reach of the ADCIRC model [BLAIN et al., 2002].

Usually, input for the ADCIRC program is a 2D unstructured mesh, values of wind and pressure. For the Louisiana coast it uses 314 nodes and works with 85% efficiency. It has a 100m resolution and in deep ocean the resolution is 50 km. In a 128 node supercomputer, it takes 1 hr of run-time to produce 1 day's forecast. It is run for dozens of simulations varying its inputs, path and strength.

Implementing ADCIRC efficiently on multiple clusters requires a sound knowledge of the ADCIRC source code. The ADCIRC source code has four main directories, namely *prep*, *metis*, *src* and *work*.

The directory "*prep*" contains the code to control domain decomposition for a parallel run. The important files in this directory are: "*adcprep.f*"- controls domain decomposition, "*prep.f*"- consists of subroutines to create input files after domain decomposition, "*adcpost.f*"- controls the merging of sub-domain output into full-domain files, "*post.f*"- merges each type of output file.

The directory "*metis*" consists of third party code written in C that is used during domain decomposition.

The directory "*src*" consists of all the code required to run ADCIRC. One important file in this directory is "*adcirc.f*", which is the top-level driver code.

In the directory "*work*" executables are built. It also contains makefiles and build-supporting files.

Prior to our work, no one has yet to run ADCIRC on multiple clusters. It may prove quite prudent during times of crisis to shorten the run-time for forecasting. This golden objective, though, can only be reached if one is able to ensure that communication overhead does not overwhelm the computational efficiency. To run ADCIRC on multiple clusters, we used MPICH-G2. MPICH-G2 allows for the joining multiple machines of homogeneous or heterogeneous architectures to run MPI applications.

To use MPICH-G2, we configured the special MPICH-G2 (Globus) on MPICH-GM (Myrinet) where MPICH-G2 is in the path before other MPICH-based implementations in Intel compilers. Then, the code was compiled with the MPICH-G2/Intel mpif90, and then a "machines" file was created that specifies the CPUs to be used. After that the binary had to be transferred to the other clusters needed for a Globus run placed in the same directory. Finally, we created a Globus "RSL" file based on the "machines" file. We will present our results in the "Initial Benchmarks for Multiple Clusters" section.

PERFORMANCE OF ADCIRC: ADVANTAGES AND DISADVANTAGES

Utilized heavily by the U.S. Army Corps of Engineers and U.S. Navy, ADCIRC has also been adopted by the national Ocean Service for U.S. East coast. It has been certified by FEMA for national Flood Insurance Program. Several other state offices have also adopted ADCIRC.

There are certain inherent advantages of the ADCIRC model. First, ADCIRC possesses a large domain. It has flexible and variable range of element areas with resolution control. ADCIRC's finite element method based solution strategy allows for a very large number of discrete points to be placed in a highly flexible and unstructured manner. It also has high resolution in coastal regions and low resolution in deep ocean. The coastlines are well defined with proper knowledge of the mainland, island and other structures- such as channels inside the mainland. Any errors in providing dynamically correct boundary conditions are more readily absorbed in the computation [WESTERINK et al., 2003].

The advantage of using ADCIRC over other storm surge models, such as SLOSH, is that input conditions can include all or part of wind stress, tides, wave stress, and river discharge to make the model output more accurate [FITZPATRICK et al., 2006].

From the computational perspective, the code is highly efficient and very large domains are possible. Loop-level optimization has been done. ADCIRC is available in single thread and parallel versions that yield the same answers to machine precision. It is capable of domain decomposition and uses distributed memory and has MPI based communication. Furthermore, ADCIRC is being constantly updated. Most areas of update relate to Civil engineering and Geo-physical aspects. Also, work on a new h-p adaptive Discontinuous Galerkin (DG) based algorithm in place of Continuous Galerkin (CG) based algorithm is currently underway. The resolution of coastal regions can also be increased to 50m for better performance.

There are a few areas of ADCIRC that need attention. The matter in which I/O, for example, affects model performance and usability is not ideal. Each sub-problem of a parallel ADCIRC simulation writes its own output files that must be globalized in a post processing step. This in and of itself does not affect performance, but it does present a user with an additional step when running ADCIRC in parallel. What does affect performance, however, is that each output file must be written sequentially. The writing of each output file blocks the progression of the simulation, so when one wants to output from the model, he must strike a balance between the run time and the output frequency/period. Current efforts are under way to provide the writing of global output files at run time, but the output scheme will nevertheless remain obstructive.

Also, in the event of restarting ADCIRC, file reading proves to be quite cumbersome. For a forecast system setup, file reading needs a significant overhaul to increase the efficiency of the setup. Currently, if ADCIRC was being run in a forecasting setup, one should only need to "cold start" ADCIRC once. If one was forcing the forecast model with wind (fort.22), for example, all subsequent simulations could then be "hot started" as each new set of wind forecasts are made available.

ADCIRC, however, requires that the wind forcing data file contains all the data from the very beginning (cold start) of the simulation. This causes the input file to grow for each subsequent simulation. This situation should be easily rectifiable because for each time step, ADCIRC only requires forcing data for the current and previous time steps.

Recently, ADCIRC has been given the ability to generate analytical winds internally using a hurricane track file consisting of a series of central pressure and coordinates. This makes ADCIRC quite beneficial when used to predict storm surges for impending hurricanes. However, ADCIRC is somewhat limited in the context of a daily forecast system during non-tropical situations.

Second, the Jacobi solver is neither parallelized plus it takes more time to converge compared to other solvers. Given that a parallelized version of the Jacobi solver exists, implementing this parallelized version should yield markedly improved results. Also, the SOR (Successive Over Relaxation) method converges faster than the Jacobi method, and a parallelized version for the SOR method exists as well.

Finally, ADCIRC lacks a comprehensive flow diagram and the parallelization of ADCIRC occurred a while back. While ADCIRC makes use of Fortran 90, the latest and newest methods found in MPI need to be exploited for maximum performance. To know effectively how to alter the MPI, one needs to know the number of nodes where communication overhead is greater than computation.

BENCMARK RESULTS FOR SINGLE CLUSTERS

To obtain a better feel for ADCIRC, several benchmarks using a different amount of processors were conducted. Benchmarking was performed on SuperMike, a 512 X2 cluster located at Louisiana State University. Each node is 3.06GHz Intel Pentium IV Xeon processor with 2GB memory per processor. The mesh sizes used in this experiment were 200K and 1000K. We ran ADCIRC on 2, 4, 8, 16, 32, 64, 96, 128, and 256 processors. The wall clock time decreased linearly as the number of CPUs was doubled. This indicates that the time required to complete an ADCIRC simulation decreases linearly as the number of processors increases. The results of these benchmarks are shown in Figure 1.

Another experiment was performed on the BlueGene/L computer which is an IBM BGL architecture with 1024 x 700MHz PPC440 processors located at Argonne National Laboratory. The

mesh size used in this experiment was 1000K. ADCIRC was run on 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024 processors. The results of this experiment are shown in Figure 2.

These benchmark results show that ADCIRC can easily scale up to 512 processors for a mesh of size 1000K and up to 96 processors for a smaller mesh, which was 200K. The largest mesh that the ADCIRC team has been working on has 2.5 million nodes, which can easily scale up to more than a thousand processors.

While not practical, these results may initially indicate that running ADCIRC on two different clusters with twice the number of nodes for a single cluster may yield better results. For example, running ADCIRC with a 1000k mesh size on two different clusters using 128 nodes should yield a markedly improved performance as opposed to running ADCIRC on 128 nodes on a single cluster.

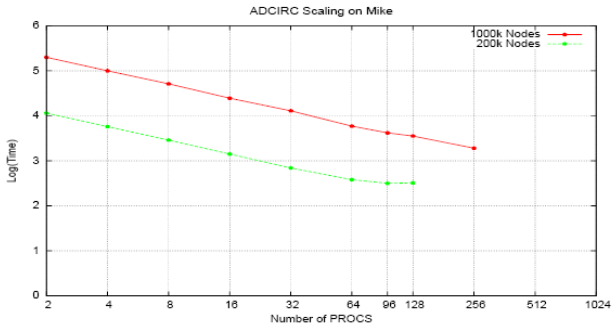

Figure 1: ADCIRC scaling on SuperMike cluster

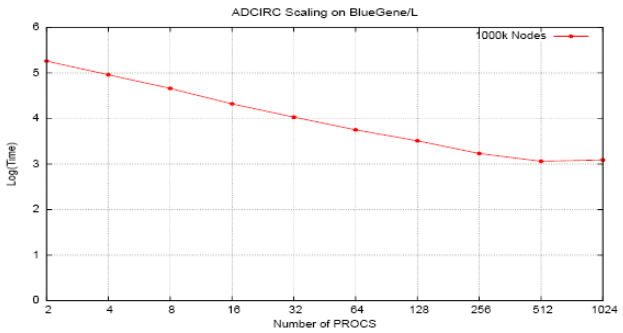

Figure 2: ADCIRC scaling on BluGene/L

GPROF RESULTS

The complexity and enormity of ADCIRC, inserting timing routines before the major routines would have proven inefficient as well as complex. We have conducted a variety of flat profiles for a 200k grid as well as the 1000k grid. We wish to focus on the gprof profiles of the 1000k grid. For columns that have the N/A categorization, the functions were not profiled by gprof.

From these tables one is able to discern a variety of interesting characteristics concerning ADCIRC. First, procedures such as hotstart_ and coldstart_ that deal with the initialization of ADCIRC seem to scale quite well. In the 4 and 32 processor scenario, these routines consumed the most time in ADCIRC. After the increase to 64 and 256 processors, these routines account for far less time of the total time spent by ADCIRC.

On the other hand, one notices procedures that were never "significant" consuming major time in the 64 and 256 processors' cases. For example, one can take itpackv_mp_ymasx2_ which represents a single Jacobi iteration that is found in itpackv.F. The Jacobi iterations account for more of the time spent by ADCIRC as the number of processors increases dramatically. Apparently, it is safe to assume that the Jacobi solver does not scale well linearly, and in this case, it may be due to the communication latency overwhelming the computational efficiency.

Table 1: Flat Profile 1000k Grid using 4 Processors

Routine	Time Spent (s)	# of Calls	% of Run Time
hotstart_	6.98	5	18.54
for_inquire	5.06	N/A	13.44
coldstart_	4.89	6,913,252	12.99
write_output_2d_	3.91	N/A	10.39
read_input_	1.99	12,407	5.29
timestep_	1.99	54	5.29
for__open_proc	1.62	N/A	4.30

Table 2: Flat Profile 1000k Grid using 32 Processors

Routine	Time Spent (s)	# of Calls	% of Run Time
hotstart_	6.96	5	17.97
coldstart_	5.95	6,938,695	15.36
for_inquire	5.34	N/A	13.79
write_output_2d_	3.42	N/A	8.83
read_input	1.76	93,689	4.54
for__open_proc	1.65	N/A	4.26
timestep__	1.58	274	4.08

Table 3: Flat Profile 1000k Grid using 64 Processors

Routine	Time Spent (s)	# of Calls	% of Run Time
Itpackv_mp_ymasx2_	1,392.88	495,964	28.64
gwce_new_	1,210.66	43,200	24.89
mom_eqs_new_nc_	685.08	43,200	14.09
timestep_	445.93	43,200	9.17
itpackv_mp_itjcg_	445.93	452,764	5.17
Itpackv_mp_unscal_	194.15	43,200	3.99
itpackv_mp_scal_	167.14	43,200	3.44

Table 4: Flat Profile 1000k Grid using 256 Processors

Routine	Time Spent (s)	# of Calls	% of Run Time
Gwce_new_	332.38	43,200	33.72
Itpackv_mp_ymasx2_	201.45	495,964	20.44
mom_eqs_new_nc_	156.56	43,200	15.88
timestep_	101.65	43,200	10.31
Pow.J	32.51	N/A	3.30
itpackv_mp_itjcg_	31.25	452,764	3.17
Itpackv_mp_scal_	28.33	43,200	2.87

Finally, there are routines that always play a major role in ADCIRC as far as time spent is concerned. An example is the timestep_ routine found in timestep.F. One particular subroutine that shows major prominence later on, gwce_new_, is a subroutine found in the timestep_ module.

From these tables, one is thus able to discern that hiding the communication latency in ADCIRC will prove to be a major benefit. By hiding the communication latency via a process found in [BARNARD et al., 1999], one should be able to enjoy the computational superiority of multiple clusters whilst avoiding the communication latency associated with multiple clusters.

BENCHMARK RESULTS FOR MULTIPLE CLUSTERS

For our initial ADCIRC runs on multiple clusters, we have used two LONI clusters. One cluster, Zeke, is located at the University of Lafayette in Lafayette, Louisiana. The other cluster used, Ducky, is located at Tulane University in New Orleans, Louisiana. These clusters are connected via a 10 Gbps network. Each LONI cluster is a 14 8-way IBM Power5 1.9GHz system running AIX5.3. Thus far, no changes have been made to the ADCIRC code. The major change was modifying the *makefile* to use the MPICH-G2 Fortran compiler. Our initial ADCIRC test runs make use of both 200k and 1000k mesh sizes on varying numbers of processors evenly divided between the two clusters.

As observed from our data, the various runs indicate the plausibility of running ADCIRC on multiple clusters. However, in the case of the 200k mesh, ADCIRC begins to scale poorly which was seen previously in single clusters as the number of processors increases. At this critical stage, it appears that the communication overhead overwhelms the computation benefit derived from the extra processors. Fruthermore, as seen in the 1000k mesh ADCIRC runs, a larger dataset allows for larger scalability. More testing is nevertheless needed as the 64 processor run (32+32) for the 1000k mesh appears to be an outlier.

Table 5: ADCIRC scaling on Zeke and Ducky

Number of Processors		Time (minutes)	
Zeke	Ducky	200k Mesh	1000k Mesh
8	0	24	81
4	4	34	108
8	8	23	55
16	16	18	40
32	32	22	51
64	64	21	26

Given the current results, an unmodified ADCIRC does not jusify the resource usage. In the case of the 200k mesh ADCIRC run, we provided eight times the resources over a single cluster to obtain an improvement of only three minutes. More work is needed to hide the communication latency of ADCIRC, which may entail modifying the source code and even some parts of the model itself.

CONCLUSIONS

In this study, we have analyzed one of the most widely used storm surge models, ADCIRC, in order to provide a set of guidelines on how its performance could be improved for faster storm surge forecasts. Our study resulted in severeal possible areas of improvement such as I/O, the solver used, parallelization technique, and distributing the application to multiple clusters. An enhanced version of ADCIRC will allow more "crunch-time" computations where a hurricane may be only 72 hours off shore. With such data at hand, catastrophes that occurred during Hurricane Katrina can be better avoided where people will have accurate knowledge to evacuate properly and know beforehand whether levies may break as seen in the aftermath of Hurricane Katrina.

ACKNOWLEDGMENTS

This work was supported by NSF grant CNS-0619843, Louisiana BoR-RCS grant LEQSF (2006-09)-RD-A-06, and CCT General Development Program. We wish to thank the ADCIRC team, especially Rick Luettich, for making the ADCIRC code available to us, and LSU CCT members including Chirag Dekate and Steven Brandt for their efforts in aiding us during this project. We also wish to thank LONI and LSU ITS for providing us with the resources to run and analyze ADCIRC.

REFERENCES

LYNCH, D.R., J.T.C. IP, NAIMIE, .CE., and F.E. WERNER, F.E., 1996. Comprehensive coastal circulation model with application to the Gulf of Maine. Continental Shelf Research, 16, 875-906.

LUETTICH, R.A., WESTERINK, J.J., SCHEFFNER, N. W, 1992. ADCIRC: an advanced three-dimensional circulation model for shelves, coasts and estuaries, Report 1: theory and methodology of ADCIRC-2DDI and ADCIRC-3DL. Dredging Research Program Technical Report DRP-92-6, U.S. Army Engineers Waterways Experiment Station, Vicksburg, MS, 137p.

LUETTICH, R.A., WESTERINK, J.J., 2004. Formulation and Numerical Implementation of the 2D/3D ADCIRC Finite Element Model Version 44.XX. August 2004.

ADCIRC Online Tour, 2006. [online]. Available from: http://veritechinc.net/products/sms_adcirc/ADCIRCshow/ADCIRCSlide2.php

WESTERINK, J.J., LUETTICH R.A.,KOLAR, R., and DAWSON C., 2003. CIRP – SMS Steering Model Workshop. ADCIRC Overview and Model Features, Presentation, 2003.

FITZPATRICK, P.,VALENTI, E., 2006. "Forecasting of Storm Surge Floods Using ADCIRC and Optimized DEMs"- Stennis Space Center, Mississippi, 2006.

BARNARD, S., SAINI, S., VAN DER WIJNGAART, R., YARROR, M., ZECHTZER, L., FOSTER, I., and LARSSON, O., 1999. Large-scale distributed computational fluid dynamics on the information power grid using globus, in 7th Symposium on the Frontiers of Massively Parallel Computation, pages 60-67, Los Alamitos, CA, 1999. IEEE Computer Society Press.

MECHOSO, C.R.,MA, C.C, Ferrara, J.D., and SPAHR, J.A., MOORE, R.W., 1993. Parallelization and Distribution of a Coupled Atmosphere–Ocean General Circulation Model. Mon.Wea. Rev., 21:2026–2076.

BLAIN, C.A., PRELLER, R.H., and RIVERA, A.P., 2002. Tidal Prediction Using the Advanced Circulation Model (ADCIRC) and a Relocatable PC-based System. Oceanography Magazine (special issue), 15 (1), 77-87, 04-FEB-2002.

MESRI, Y., DIGONNET. H., and GUILLARD. H., 2005. Mesh Partitioning for Parallel Computational Fluid Dynamics Applications on a Grid, in Finite Volumes for complex applicationsIV. Hermes Science Publisher, p. 631-642