

# An educational interactive numerical model of the Chesapeake Bay<sup>☆</sup>

Jessica R. Crouch<sup>a,\*</sup>, Yuzhong Shen<sup>b</sup>, Jay A. Austin<sup>c</sup>, Michael S. Dinniman<sup>d</sup>

<sup>a</sup>*Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA*

<sup>b</sup>*Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529, USA*

<sup>c</sup>*Large Lakes Observatory, University of Minnesota, Duluth, MN 55812, USA*

<sup>d</sup>*Center for Coastal and Physical Oceanography, Old Dominion University, Norfolk, VA 23529, USA*

Received 24 July 2006; received in revised form 23 March 2007; accepted 26 March 2007

---

## Abstract

Scientists use sophisticated numerical models to study ocean circulation and other physical systems, but the complex nature of such simulation software generally make them inaccessible to non-expert users. In principle, however, numerical models represent an ideal teaching tool, allowing users to model the response of a complex system to changing conditions. We have designed an interactive simulation program that allows a casual user to control the forcing conditions applied to a numerical ocean circulation model using a graphical user interface, and to observe the results in real-time. This program is implemented using the Regional Ocean Modeling System (ROMS) applied to the Chesapeake Bay. Portions of ROMS were modified to facilitate user interaction, and the user interface and visualization capabilities represent new software development. The result is an interactive simulation of the Chesapeake Bay environment that allows a user to control wind speed and direction along with the rate of flow from the rivers that feed the bay. The simulation provides a variety of visualizations of the response of the system, including water height, velocity, and salinity across horizontal and vertical planes.

© 2007 Elsevier Ltd. All rights reserved.

**Keywords:** Modeling; Simulation; Visualization; Oceanography; Education; Interactive; Chesapeake Bay

---

## 1. Introduction

Geoscientists use numerical models to study a variety of complex physical systems such as ocean circulation, weather development, and landform evolution. Researchers investigate scenarios with these models by specifying boundary conditions

that describe the external influences on a system and compute predicted values for quantities of interest based on the model's equations. By carefully controlling the forcing, an investigator can isolate the role of specific forcing parameters. Numerical simulations therefore enable scientists to make forecasts and perform virtual experiments that are impossible to perform in reality.

The ability to investigate the role of specific forcing mechanisms should, in principle, make numerical models ideal teaching tools, allowing either an instructor to demonstrate a particular

---

<sup>☆</sup> Code available from server at <http://www.iamg.org/CGEditor/index.htm>

\*Corresponding author. Tel.: +1 757 683 6001.

E-mail address: [jrcrouch@cs.odu.edu](mailto:jrcrouch@cs.odu.edu) (J.R. Crouch).

phenomenon, or students to engage in active learning by using the model themselves. However, three major factors have kept these models out of the classroom. First, the complexity of these models typically puts them out of the reach of all but a small handful of researchers who have devoted large amounts of effort to understanding the code and developing visualization routines for the output. Second, models are typically run in an asynchronous fashion, with the investigator first setting up the conditions for the model, then running the model, and finally visualizing and interpreting the results. This can make it difficult for students to generate an “intuition” about the behavior of a system. Finally, until recently, these models have required computing resources unavailable in a typical classroom.

While the expertise of professional scientists is needed to design and implement numerical models, members of the larger community can improve their understanding of the environment through interaction with simulation software. In communities around the Chesapeake Bay, in particular, there is broad interest in human impacts on the health of the bay’s complex ecosystem (Boesch et al., 2001; Officer et al., 1984). The environmental issues surrounding the Chesapeake Bay make it an especially important topic for public education and discussion for ecological, economic, and political reasons.

The Chesapeake Bay simulation program presented here is the product of a pilot project intended to demonstrate the feasibility of making the capabilities of the Regional Ocean Modeling System (ROMS), one of the ocean circulation research codes, accessible to non-expert members of the community. The goal was to create a simulation program that would be both scientifically sophisticated and user-friendly. The design and implementation of this program involved re-engineering ROMS to run interactively instead of in batch mode, creating a new user-friendly graphical user interface, and integrating real-time three-dimensional data visualization routines. The resulting simulation allows a user to control wind speed and direction along with the rate of flow from the rivers that feed the bay. The program responds to changing conditions, takes tidal effects into account, and provides visualizations of water height, velocity, and salinity across the bay’s surface and in cross-section.

The remainder of this paper is organized as follows. Section 2 surveys previous work in education-oriented geoscience simulation, and Section 3 describes the software architecture of the Chesapeake Bay simulation program. Section 4 presents case studies of Chesapeake Bay phenomena that can be simulated, and Section 5 concludes this paper and discusses possibilities for extending the current proof-of-concept program.

## 2. Previous work

Science education researchers report that computer simulations with manipulable models can foster inquiry-based learning (Windschitl, 2000). Such programs allow students to ask questions, form hypotheses, and perform experiments in a simulated environment. This activity facilitates exploration and discovery, allowing students to practice the scientific method instead of merely reading or hearing about the scientific work of others. Although not a replacement for laboratory or field exercises, simulations allow supplemental experiments to be performed quickly, requiring less equipment and student supervision than traditional experiments. Topics like Newtonian mechanics lend themselves to simple equation-based simulation, and a wide variety of such education-oriented simulation programs are available (de Jong et al., 1999; Dede et al., 1996; Härtel, 2000). Simulations for more complex phenomena can be equally valuable but more difficult to implement.

In the realm of geoscience education, a variety of instructional software is available, but reports of the educational use of numerical simulation programs are relatively rare. Geoscience software that has been effectively used for instruction includes role-playing games in which students play the part of a scientist investigating a virtual world (Saini-Eidukat et al., 2002; Renshaw and Taylor, 2000), and data analysis and visualization programs that allow users to explore large sets of collected or simulated data. For example, Ramasundaram et al. (2005) presented software that lets users interact with water table data collected over a six year period in a 42-ha flatwood forest. More recently, Winn et al. (2006) presented software that enabled students to visualize water movement and salinity data computed for Puget Sound using the Princeton Ocean Model (Blumberg and Mellor, 1987). This instructional program allows exploration of the saved results of a

simulation, and lets users select one of two data files that were generated using different forcing conditions. The program does not allow a user to specify new forcing conditions and compute new simulation results. Similarly, the EdGCM project (Chandler et al., 2003) provides a graphical user interface for visualizing data computed by a global climate model, but the program does not provide users with an ability to change forcing conditions and compute the impact of such changes.

The distinction between interactive exploration of recorded data and interactive numerical simulation is an important one. Both can be interesting and educationally valuable, but in the first case the scenarios that a user can investigate are strictly limited to the simulations that were recorded by the software's authors. In contrast, a simulation program that gives the user control over model forcing variables and then computes the effects of the chosen settings provides access to a virtually limitless number of scenarios. A user who is given control over the forcing conditions can engage in genuine scientific inquiry by asking new questions, forming hypotheses, and running simulation experiments.

Few interactive numerical simulations exist that are appropriate for non-expert users. Published reports of educational simulation software indicate that the models employed tend to be based on less complex governing equations than those in research-oriented simulations. An example is the climate model presented by Bice (2001). This teaching model represents the earth's climate using two reservoirs of thermal energy: one for the atmosphere and one for the earth's surface. This model simulates gross climate change at a high level of abstraction, but does not represent the spatial distribution of temperature. Another example of an educational simulation is WILSIM (Luo et al., 2004), a software program that models landform evolution due to the erosion, diffusion, and deposition of soil over time. WILSIM uses a cellular automata algorithm to represent the movement of water and soil across a virtual landscape. The cellular automata model succeeds in simulating large scale effects, but has some limitations compared to the more computationally intensive approach of solving the hydrodynamic equations. The authors report the simulation "is based on 'very simple approximations intended to capture the synoptic effects of fluvial processes' (Chase, 1992)." The authors further warn that the simplifications

lead to incorrect model behavior such as "sediment accumulation at the bottom of the channels in rugged terrains, which is contrary to real world situations." In general, model simplifications ease implementation and reduce computational power requirements, but also limit a simulation's reliability.

An example of an equation-based, interactive, and education-oriented geoscience simulation program is the sea wave forecasting program by Whitford (2002a, b). This program allows a user to control model variables such as wind speed and duration and computes new simulation results for user-specified settings. The program consists of a suite of MATLAB functions, so some facility using the MATLAB command line environment is required in order to use this program. While this is a reasonable expectation for the university science students who form the program's intended user base, it does preclude use by more casual users and anyone who lacks a MATLAB installation.

In summary, most education-oriented geoscience simulation programs use relatively simple models partially because implementation of complex research grade numerical models requires an unreasonable amount of development effort for educational software. The alternative approach taken by some authors has been to generate data files using a research-oriented numerical model and to provide students with a data visualization program that allows them to interactively explore the saved simulation results. The available educational simulation programs therefore tend to work interactively with a simplified model or be limited by static data files. The Chesapeake Bay simulation program presented in this paper was developed using a different approach. Code for a graphical user interface and visualization functions was integrated with the code for a research grade ocean circulation model. This produced a user-friendly program that allows users to set forcing conditions and interactively compute new simulation results by engaging a sophisticated numerical model. One of the main objectives of this project was to find a balance between spatial/temporal resolution and execution speed. While the system is not resolved at the resolution that a research-quality implementation of the Chesapeake might be, it still captures the essential character of estuarine circulation and runs quickly enough that it can retain the interest of a casual user.

### 3. Simulation architecture and implementation

The three functional units that compose the Chesapeake Bay simulation program are the graphical user interface, the numerical ocean circulation modeling code, and the visualization routines. The data flow between these is illustrated in Fig. 1. A description of the Chesapeake Bay is provided as input data to the simulation. A description of each of the simulation components follows.

#### 3.1. Regional Ocean Modeling System (ROMS)

ROMS is an ocean circulation modeling program created and supported by members of the oceanographic research community (Shchepetkin and McWilliams, 2005). The source code for ROMS version 2.1 was the starting point for the development of the Chesapeake Bay simulation program. This research grade circulation model was selected as the computational engine of the simulation because it supports a wide variety of circulation model properties. A subset of ROMS capabilities is accessible in the first version of the interactive Chesapeake Bay simulation, but the expansion of this feature set will be straightforward since ROMS is already capable of modeling properties such as oxygen concentrations and pollutant distributions.

ROMS uses the hydrostatic primitive equations to model ocean circulation. These equations are a simplification of the Reynolds-averaged Navier–

Stokes equations and describe fluid flow on a sphere, under the assumption that the depth of the fluid layer is much smaller than the sphere's radius. ROMS uses a finite difference type of approach to solve the hydrostatic primitive equations on a grid defined by terrain following coordinates called sigma coordinates. This form of grid allows the topography of the ocean floor and coast to be directly incorporated in a model. As input, ROMS requires parameters for the grid and a set of forcing conditions. Forcing conditions can include time and location dependent functions for variables such as wind speed and direction, atmospheric temperature, precipitation, river flows, tides, and pollution sources. During the solution phase ROMS computes distributions for these and other variables throughout the three-dimensional volume of a model at each time step.

ROMS is written in Fortran 90 and consists of more than 340,000 lines of source code. In order to produce compact, efficient executables from a large code base, ROMS makes extensive use of compiler directives to exclude portions of the code that are unnecessary for specific types of input models. This code structure makes editing and recompilation of the code necessary for each model variation. Furthermore, ROMS is designed to run in batch mode, managing all model input and solution output through files. Interactive adjustment to simulation variables is not supported. Visualizations of output data, if needed, are generally created by

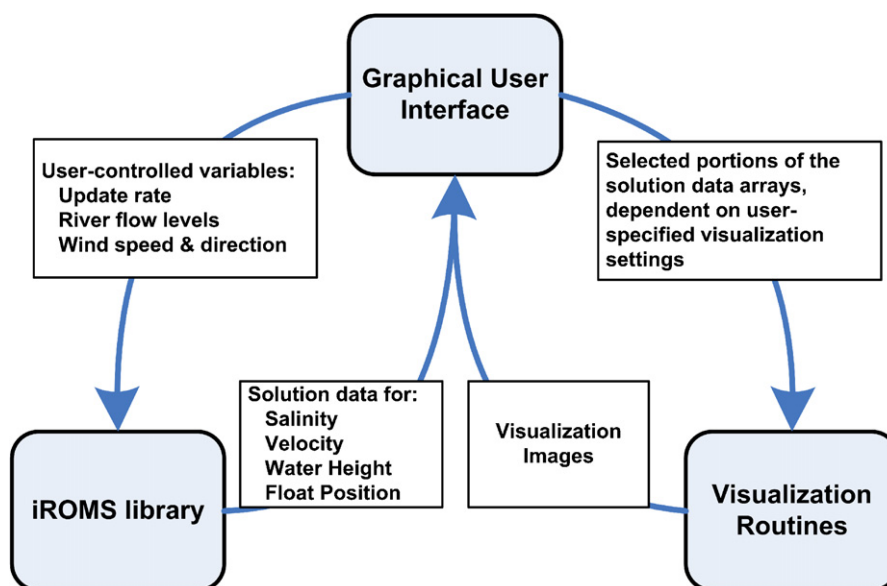


Fig. 1. Software architecture of interactive Chesapeake Bay simulation system.

researchers during a post-processing step using secondary software such as MATLAB. Therefore, ROMS provides researchers with low-level control and great modeling flexibility but lacks the user-friendly facilities needed in software intended for novice users. Indeed, graduate level oceanographers often spend weeks to months learning to use ROMS. The requisite level of user sophistication precludes ROMS use by most students and casual users.

Modifications to ROMS were necessary in order to make its powerful numerical modeling capabilities more accessible. The outermost loop in the ROMS source code advances the simulation time and calls routines to set up and solve equations for the current time step. The primary modification to ROMS was the removal of this time-stepping loop and the addition of new top level functions to manage simulation time increments, update the forcing conditions at the beginning of each update, and pass the computed solution data to the visualization routines at the end of each update. The time step size used by ROMS cannot be controlled directly by a user because it impacts the numerical stability of the solution routines. However, ROMS can integrate the results from multiple time steps so that effects at different time scales can be examined. User changes to the simulation update rate are implemented as changes to the integration period. Originally written as a standalone executable program, the modified ROMS code was compiled into a linkable library called iROMS (interactive ROMS).

### 3.2. Chesapeake Bay model

The iROMS code used for the Chesapeake Bay model was configured by enabling the algorithms for computing salinity levels, tracking floating objects, using a K-profile parameterization for vertical mixing (Large et al., 1994), controlling tides at the open boundary, and managing point sources of momentum, temperature, and salinity. A computational grid was defined to represent the entire Chesapeake Bay, stretching approximately 200 miles from Havre de Grace, Maryland, to Cape Henry, Virginia. Portions of several rivers feeding the bay are represented in the grid, including the Susquehanna, Potomac, James, Rappahannock, and York Rivers.

The dimensions of a computational grid are an important factor in the computational complexity

of the iROMS solution algorithm, so the desired simulation update rate influenced the design of a grid for the Chesapeake Bay model. The goal for the Chesapeake Bay grid was to be small enough that the simulation would complete a solution step in less than a second while running on a PC and yet be large enough to demonstrate interesting effects in the bay. The chosen grid dimensions are  $20 \times 50 \times 10$  (east–west  $\times$  north–south  $\times$  vertical layers). Latitude, longitude, and altitude coordinates are stored for each grid vertex. Research versions of the Chesapeake Bay model typically have higher grid resolution and require offline computation that can take hours to complete on a cluster computer. With future advances in computational speed, the grid resolution for the Chesapeake Bay could easily be increased. User-adjustable grid resolution is part of planned future work. Adjustable grid resolution would allow students to examine how the computed circulation features vary with grid size and how the required computation time is impacted by the resolution.

### 3.3. Graphical user interface

A user interface was implemented in C++ using the Fast Light Toolkit (fltk), an open source cross platform library. The user interface code contains the program's main event loop that controls the simulation's execution and flow of data at the highest level. Whenever a user changes the setting of one of the on-screen controls, the event loop receives notification of the change and updates the parameters passed to iROMS at the beginning of the next solution step. Calls to iROMS are implemented by spawning new threads. The multi-threaded nature of the program enables the user interface to provide smooth interactivity without blocking while waiting for iROMS to finish computing a solution step. Another benefit of the multi-threaded implementation is that the simulation can take advantage of a dual processor PC by continuously running iROMS calculations on one processor while using the other processor for interface and graphics functions.

The interface provides two categories of controls: those that affect the numerical model and those that affect the visualization of computed results. Controls affecting the numerical model include a selection box for the update rate, sliders to set river flow rates, and a slider and dial for setting the wind speed and direction. The interface displays the



simulation time, updating it after each solution time step. An update frequency in the range of 1–24 h can be selected, influencing the degree of variability due to tidal effects a user will notice. The rate of flow can be set for the three largest rivers feeding the bay: the Susquehanna, Potomac, and James Rivers. For this simulation the wind is assumed to be constant across the surface of the bay, and users may select a single wind speed and direction.

Controls affecting the visualization of the computed results include check boxes to select a particular variable for visualization and a slider to select the depth of the grid layer to visualize. Either the water's salinity, velocity, height level, or the position of floating markers can be selected for visualization. Since the computed results are three-dimensional, plotting data on a map of the bay requires a two-dimensional subset of the data be selected. The depth slider allows users to select any of the grid's 10 vertical layers for visualization. The user interface is shown in Fig. 2.

### 3.4. Visualization routines

Visualization routines were written using OpenSceneGraph, an open source cross platform gra-

phics library. Plan view and depth view windows provide orthogonal visualizations of the bay. The plan view displays a map of the bay and surrounding coastal areas, complete with rivers and nearby cities. The latitude/longitude coordinates associated with computational grid vertices correspond to  $(x,y)$  points on the map plot. Data values are assigned to map locations based on these coordinates and are linearly interpolated between vertices. The following types of visualizations can be selected for the plan view.

*Water height* is a scalar quantity represented by applying color mapping to the bay. With a 1 h update rate, tidal variations in water height are evident using this visualization. The water height visualization is displayed in Fig. 3(a).

*Salinity* is another scalar quantity represented via color mapping. Variations in fresh water flow from the rivers feeding the bay cause pronounced changes in salinity. Salinity visualizations are shown in Fig. 6 for the example simulation described in Section 4.3.

*Velocity* vectors of unit length are drawn on the bay to show the direction of water flow. The vectors are color mapped to indicate the magnitude of the velocity. This alternative to using arrow length to

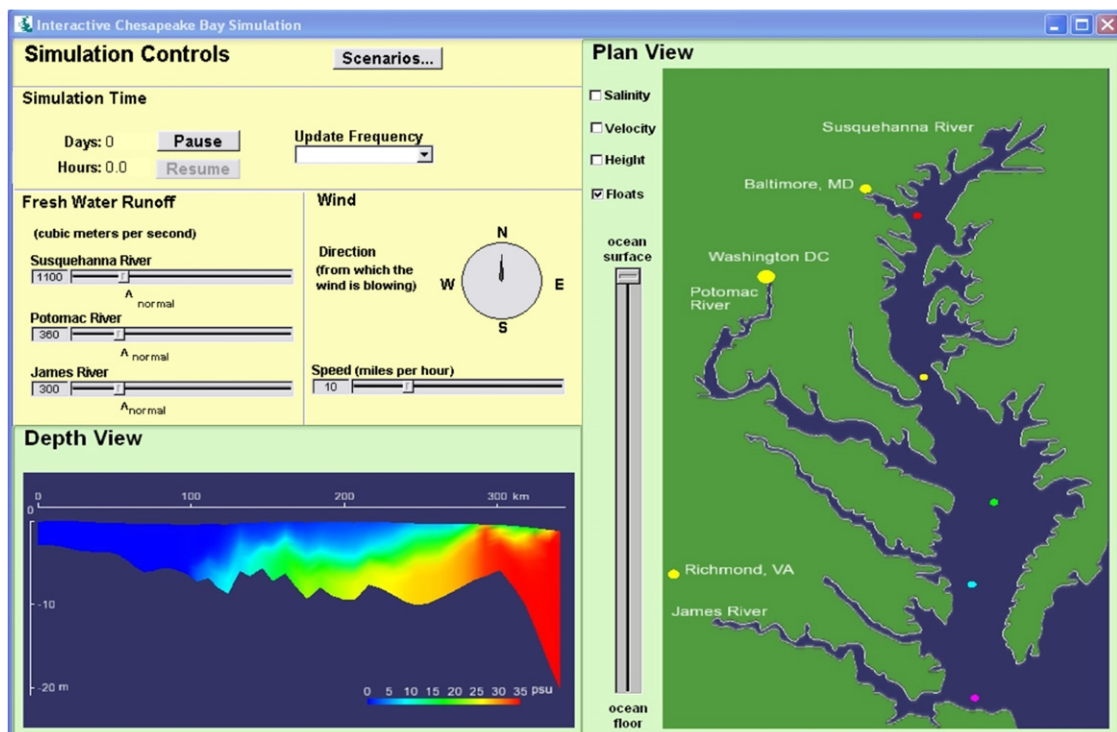


Fig. 2. Interface for Chesapeake Bay simulation program.

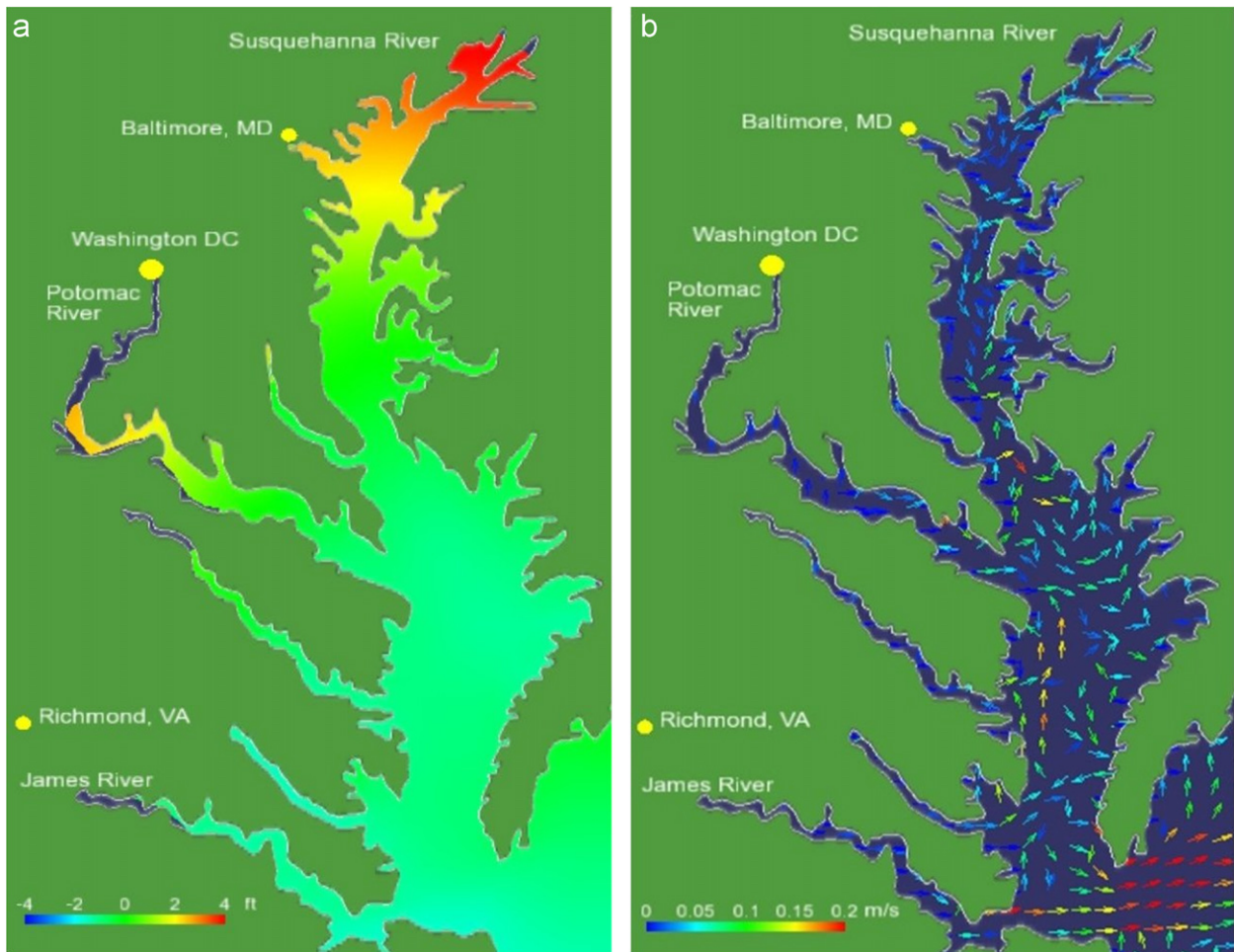


Fig. 3. Plan views. (a) Water height levels displayed via colormapping. (b) Velocity direction vectors with colormapped magnitudes.

indicate vector magnitude eliminates the problem of large magnitude vectors obscuring neighboring vectors and cluttering the display. Changes in the wind tend to affect water velocity. This visualization is shown in Fig. 3(b).

*Particle tracking* allows visualization of water flow over time. Five floating virtual particles are dropped onto the surface of the bay every 20 simulation days. The particle positions are tracked in iROMS and plotted at each time step. This visualization shows how winds, rivers, and tides can affect the path of floating objects over a multi-week time scale. Particle markers can be seen in the plan view portion of Fig. 2.

The depth view window provides a visualization of a cross-section of the bay that follows a center channel running roughly north to south. The bay's geometry in this view accurately represents the topography of the floor and the water's time and

location dependent height variations. The salinity distribution on the cross-section is displayed via color mapping, as shown in the lower left corner of Fig. 2.

### 3.5. Performance

The Chesapeake Bay simulation program runs at interactive rates on currently available PC hardware. On a 3.2 GHz dual Intel Xeon processor PC, 25 simulation days pass per minute of user time. On a laptop PC with a single 1.6 GHz Intel processor, the simulation runs at a rate of 11 simulation days per minute. This means that when the visualization is set to update at the end of each hour of simulation time, the results are refreshed every 0.10–0.23 s.

The program is currently available for download from the authors' web site at <http://www.d.umn.edu/~jaustin/CHIMP.html> as a self-installing

executable program that will run under the Microsoft Windows operating system. Because the program contains no platform dependent source code, there is no obstacle to compiling the simulation program for other operating systems or other types of computer hardware.

#### 4. Case studies

The Chesapeake Bay simulation program is user-friendly and appropriate for a variety of users including middle school, high school, and university students, museum visitors, and other members of the community. It is capable of demonstrating the basic features of the bay, including layered circulation, wind mixing, rotationally controlled flow, wind-driven flow, and tides. Three simulated scenarios that illustrate interesting Chesapeake Bay phenomena are described in the following sections. Specific user instructions for replicating

each scenario are included, along with a description of the effects observed in the simulation.

##### 4.1. Case study 1: normal estuarine salinity and circulation

Estuaries such as the Chesapeake Bay are continuously fed by a source of fresh water at their head. The result is that normal estuarine circulation consists of upper layers of relatively fresh water flowing out of the bay and lower layers of heavier, saltier water flowing up into the bay. Due to mixing effects, water closer to the head of the bay is less salty than water near its mouth.

*User instructions:* Set the update rate to 1 h, and set the Susquehanna River flow to a normal or higher than normal level. Turn the wind off. Observe changes in the salinity over the next 90 simulation days. Next, set the update rate to 12 h,

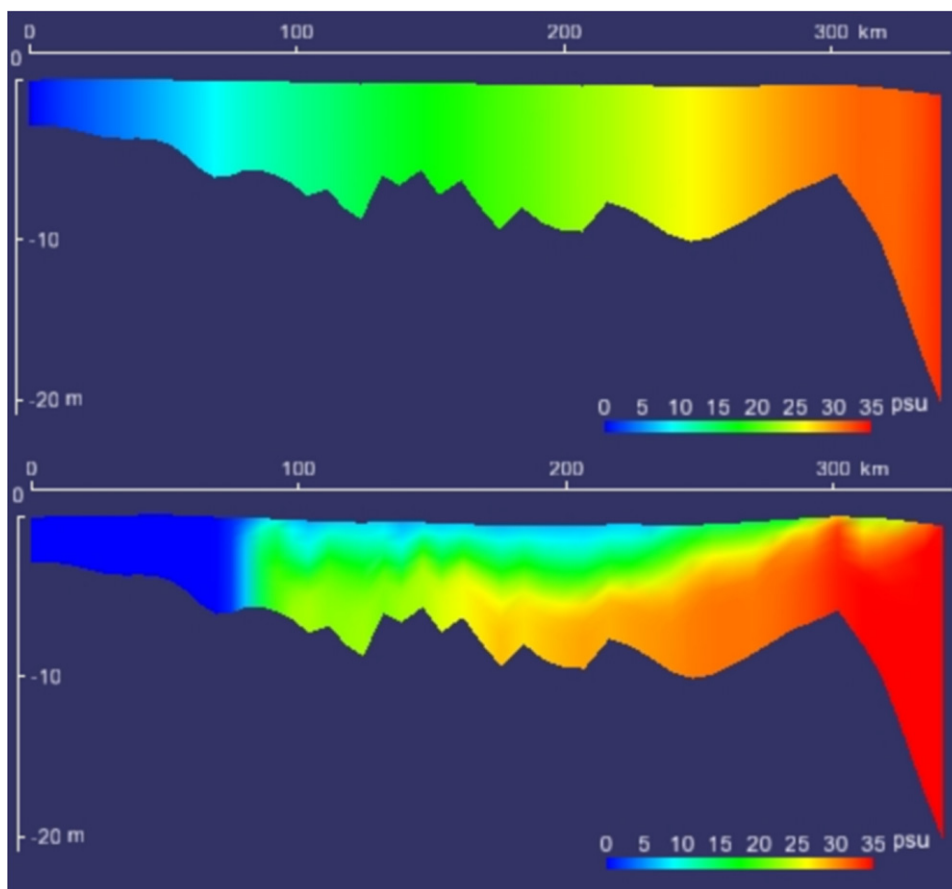


Fig. 4. Top: Initial salinity distribution in bay at beginning of simulation. Bottom: Salinity distribution in bay after stratification has formed.



and examine the velocity vector visualization for the surface and bottom layers of the bay.

**Observed effects:** Throughout the simulation the plan view shows low salinity at the head of the estuary and higher salinity at the mouth, where oceanic water is being pulled into the estuary. The cross-section initially shows homogeneous salinity through vertical water columns, with fresher water near the head of the bay and saltier water near the mouth. Without wind to mix the water, the denser, saltier water sinks to the bottom and salinity stratification becomes apparent in the depth view. The depth view also shows a fresh water/salt water boundary near the head of the Susquehanna that fluctuates with the tides. Fig. 4 shows snapshots of the salinity distribution that evolves in the depth view during this simulation. When the update rate is changed to 12 h so that the simulation results are averaged over a tidal period, the bay's two layer flow becomes evident. The velocity vectors along the bottom of the bay show water flowing toward the bay's head while the vectors in the surface layer show water flowing toward the bay's mouth. This sub-tidal two layer flow is an important feature of stratified estuaries like the Chesapeake Bay.

#### 4.2. Case study 2: wind effects

During the summer the Chesapeake Bay has few strong wind events, leading to stratification as seen in the first case study. The isolation of bottom layer of water from the surface combined with bacterial decomposition of detritus results in oxygen depletion

at the bottom of the bay. A severe storm will mix the water column, improving oxygenation. Storm winds from the south can also affect water height near Maryland by blowing water into or out of the north end of the bay.

**User instructions:** Set the update rate to 1 h. Turn the wind off for 90 days, then turn on a strong north wind. Observe changes in the bay's conditions using the salinity, height, velocity, and floating particle visualizations.

**Observed effects:** The water's surface height is visibly depressed when the north wind is turned on, and the salinity stratification is lost as the wind increases the mixing between the upper and lower layers of water. The result is that vertical water columns take on more uniform salinity values, as shown in Fig. 5.

#### 4.3. Case study 3: plumes

Because of the direction of the earth's rotation, water leaving the Chesapeake Bay tends to flow along the Virginia Beach coast and entering water tends to flow along the east side of the bay. This leads to a plume of high salinity along the southeast coast of the bay. When a north wind blows, the plume tightens against the shore due to Ekman transport in the surface layer. Ekman transport describes the tendency of surface water to travel to the right of the direction of the wind in the northern hemisphere. In the case of a north wind, surface waters in the central bay move west, flattening the plume against the shore. Conversely, a south wind

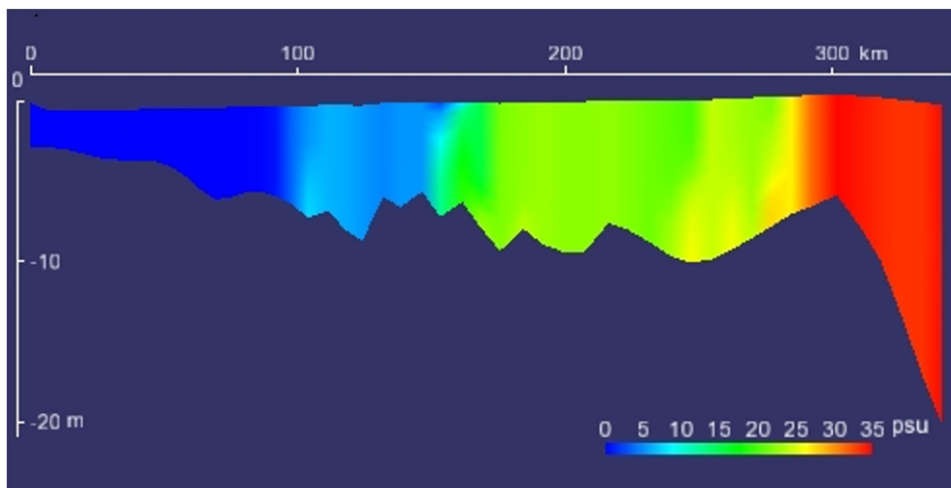


Fig. 5. This cross-section of Chesapeake Bay during a strong north wind shows reduced water height at bay's head due to wind pushing water out. Salinity stratification has been eliminated by wind mixing water.

will cause the plume to spread out into the bay as the surface water moves east.

*User instructions:* Start with the wind off and the Susquehanna flow set low for 30 simulation days. Then set the Susquehanna flow to high for 60 days. Finally, turn on a gentle wind from the north and observe changes in the salinity distribution.

*Observed effects:* When the increased volume of fresh water from the Susquehanna reaches the mouth of the bay, it turns to the right and moves south along the coast as a plume. When the north wind is turned on, the plume tightens up against the shore below the mouth of the Potomac, as shown in Fig. 6. If the experiment is repeated using a south wind, the plume diffuses into the bay.

## 5. Conclusions and future work

As demonstrated through the three example scenarios, the Chesapeake Bay simulation program accounts for a variety of complex influences on the bay, including variations in tides, winds, and river flows over time and the effects of the earth's rotation. This level of complexity is possible because a research-grade numerical circulation modeling code is employed as the computational engine of the simulation. In addition to providing instructional software for the Chesapeake Bay, the more general achievement of this simulation development has been to demonstrate that it is possible to run small numerical models on PCs in an interactive manner. Real-time interaction with an interesting

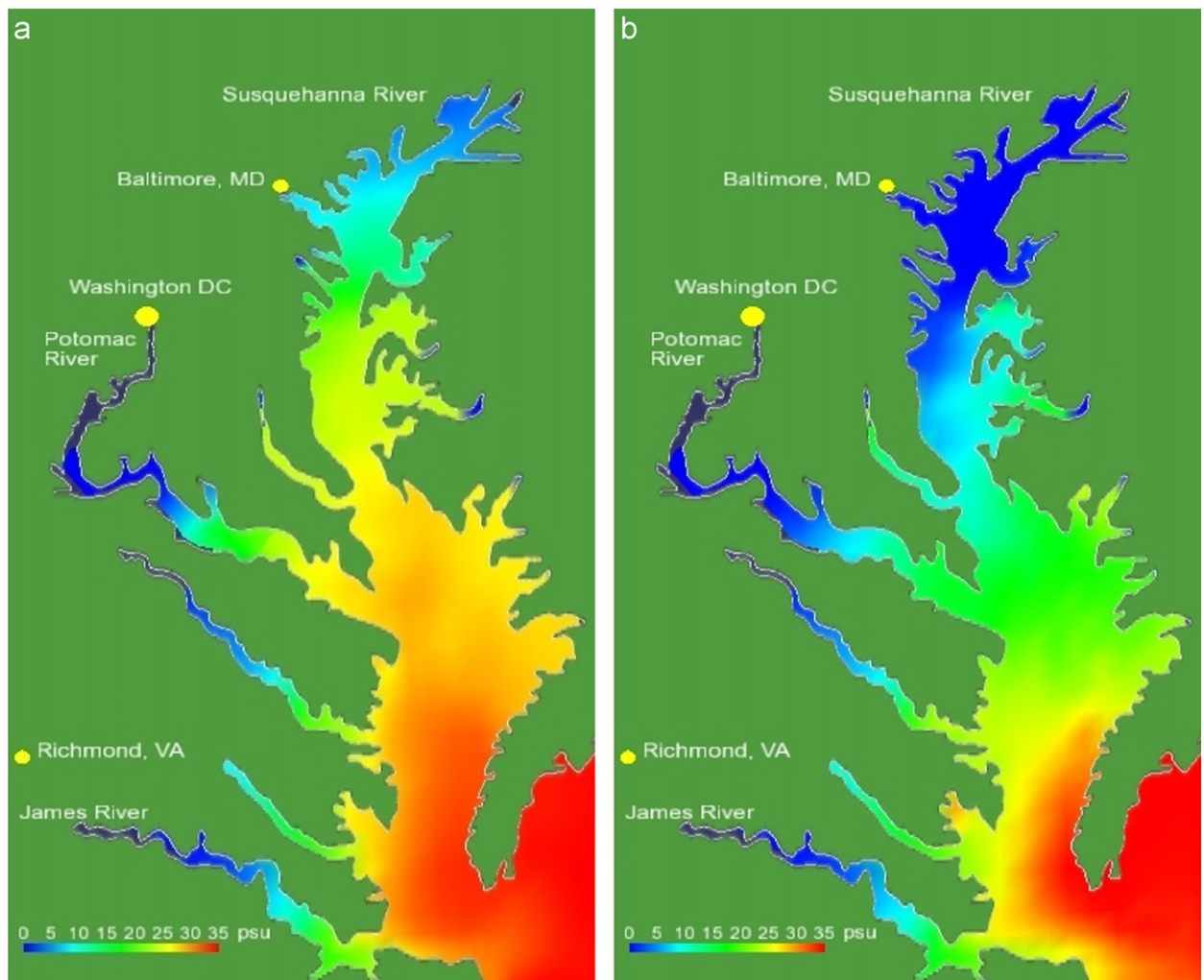


Fig. 6. (a) Surface salinity of Chesapeake Bay after weeks without wind and rain. (b) A high salinity plume along southeast coast of bay develops when scenario (a) is affected by a storm, simulated by increased river flow and wind.

numerical model marks an important step beyond previous work that provided interactive visualization of stored simulation data. This demonstration of true interactive simulation should motivate future efforts to create educational software by packaging scientific research code in an accessible manner.

The Chesapeake Bay simulation program has received positive reviews from students, educators, and others after demonstrations that have included an Earth Day event at Old Dominion University, an Oceanography Day presentation at a primary school, and other formal and informal presentations with students ranging from second grade to undergraduate university students. More formal evaluation of the program's educational utility is planned after the development of curriculum materials to accompany the program is complete. The possibility of placing the program in a marine science museum exhibit is also being investigated.

Development efforts for the Chesapeake Bay simulation program are proceeding in two directions. First, there is no reason that the simulation program needs to be limited to the Chesapeake Bay location. ROMS research models have been developed for a variety of bodies of water, and one goal is to make more of these models accessible through an interactive simulation program. Instead of writing and compiling a new program for each location, the plan is to encapsulate all the pertinent information about a model's geometry, grid characteristics, and forcing conditions (e.g., the number and names of rivers feeding the bay) in a data file that can be loaded using a general purpose interface program. The loaded data will be used to configure an array of user interface controls and the visualization options. A substantial amount of software design and engineering will be required to adapt the Chesapeake Bay simulation program to accommodate this level of generality, but the end result will be a much more flexible simulation tool. After the model data file format is established and published, any researcher who develops a ROMS model will be able to adapt it for use in the interactive simulation environment. An online repository of downloadable models is envisioned for the future.

The second development direction is expansion of the set of ROMS features that are accessible through the interactive environment. Much of the functionality available in ROMS remains untapped by the Chesapeake Bay simulation program. Visualization of pollution and oxygenation distributions

will be added, along with user controls to govern the number, location, and type of pollution sources and the computational grid dimensions. A new type of visualization is planned that will allow a user to simulate a real world experiment by dropping virtual buoys into the water and plotting the data that is measured at those locations. These future developments will make the interactive simulation environment an even more powerful educational tool.

## Acknowledgments

This project was sponsored by the Virginia Governor's Research Initiative through the Office of Research at Old Dominion University's Office of Research. The authors thank John Klinck, Lee Belfore, and Elizabeth Smith for their assistance.

## References

- Bice, D.M., 2001. Using STELLA models to explore the dynamics of earth systems: experimenting with earth's climate system using a simple computer model. *Journal of Geoscience Education* 49 (2), 170–181.
- Blumberg, A.F., Mellor, G.L., 1987. A description of a three-dimensional coastal ocean circulation model. In: Heaps, N.S. (Ed.), *Three-Dimensional Coastal Ocean Models*. American Geophysical Union, Washington, DC, pp. 1–16.
- Boesch, D.F., Brinsfield, R.B., Magnien, R.E., 2001. Chesapeake Bay eutrophication: scientific understanding, ecosystem restoration, and challenges for agriculture. *Journal of Environmental Quality* 30, 303–320.
- Chandler, M., Shopsis, M., Richards, S., 2003. EdGCM: real-time global climate modeling research for the classroom. *Geological Society of America* 35 (6), 118.
- Chase, C.G., 1992. Fluvial landsculpting and the fractal dimension of topography. *Geomorphology* 5, 39–57 (special issue on fractals).
- de Jong, T., Martin, E., Zamarro, J.-M., Esquembre, F., Swaak, J., van Joolingen, W.R., 1999. The integration of computer simulation and learning support: an example from the physics domain of collisions. *Journal of Research in Science Teaching* 36 (5), 597–615.
- Dede, C.J., Salzman, M.C., Loftin, R.B., 1996. The development of a virtual world for learning Newtonian mechanics. In: MHVR '94: Selected Papers from the First International Conference on Hypermedia, Multimedia, and Virtual Reality: Models, Systems, and Applications. Springer, London, UK, pp. 87–106.
- Härtel, H., 2000. xyZET: a simulation program for physics teaching. *Journal of Science Education and Technology* 9 (3), 275–286.
- Large, W.G., McWilliams, J.C., Doney, S.C., 1994. Oceanic vertical mixing: a review and model with a nonlocal boundary layer parameterization. *Reviews of Geophysics* 32, 363–403.
- Luo, W., Duffin, K.L., Peronja, E., Stravers, J.A., Henry, G.M., 2004. A web-based interactive landform simulation model (WILSIM). *Computers & Geosciences* 30, 215–220.

- Officer, C.B., Biggs, R.B., Taft, J.L., Cronin, L.E., Tyler, M.A., Boynton, W.R., 1984. Chesapeake Bay anoxia: origin, development, and significance. *Science* 223 (4631), 22–27.
- Ramasundaram, V., Grunwald, S., Mangeot, A., Comerford, N.B., Bliss, C.M., 2005. Development of an environmental virtual field laboratory. *Computers & Education* 45 (1), 21–34.
- Renshaw, C.E., Taylor, H.A., 2000. The educational effectiveness of computer-based instruction. *Computers & Geosciences* 26 (6), 677–682.
- Saini-Eidukat, B., Schwert, D.P., Slator, B.M., 2002. Geology explorer: virtual geologic mapping and interpretation. *Computers & Geosciences* 28 (10), 1167–1176.
- Shchepetkin, A.F., McWilliams, J.C., 2005. The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modeling* 9, 347–404.
- Whitford, D.J., 2002a. Teaching ocean wave forecasting using computer-generated visualization and animation—part 1: sea forecasting. *Computers & Geosciences* 28 (4), 537–546.
- Whitford, D.J., 2002b. Teaching ocean wave forecasting using computer-generated visualization and animation—part 2: swell forecasting. *Computers & Geosciences* 28 (4), 547–554.
- Windschitl, M., 2000. Supporting the development of science inquiry skills with special classes of software. *Educational Technology Research and Development* 48 (2), 81–95.
- Winn, W., Stahr, F., Sarason, C., Fruland, R., Oppenheimer, P., Lee, Y.-L., 2006. Learning oceanography from a computer simulation compared with direct experience at sea. *Journal of Research in Science Teaching* 43 (1), 25–42.