INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **42**:549–580 (DOI: 10.1002/fld.531)

Development of a parallel storm surge model

Justin R. Davis*,† and Y. Peter Sheng‡

Department of Civil and Coastal Engineering, University of Florida, Gainesville, Florida, 32611, U.S.A.

SUMMARY

A new parallel storm surge model, the Parallel Environmental Model (PEM), is developed and tested by comparisons with analytic solutions. The PEM is a 2-D vertically averaged, wetting and drying numerical model and can be operated in explicit, semi-implicit and fully implicit modes. In the implicit mode, the propagation, Coriolis and bottom friction terms can all be treated implicitly. The advection and diffusion terms are solved with a parallel Eulerian–Lagrangian scheme developed for this study. The model is developed specifically for use on parallel computer systems and will function accordingly in either explicit of implicit modes. Storm boundary conditions are based on a simple exponential decay of pressure from the centre of a storm. The simulated flooding caused by a major Category 5 hurricane making landfall in the Indian River Lagoon, Florida is then presented as an example application of the PEM. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: message passing; Beowulf cluster; storm surge; Indian River Lagoon; Eulerian–Lagrangian method

1. INTRODUCTION

As the population of the world's coastal communities grows increasingly larger, more attention is now being paid to the health of the surrounding estuarine and coastal waters. For many years scientists have conducted numerous laboratory and field studies to better understand the complex physical and ecological processes of these aquatic environments. Using the knowledge gleaned from these studies, it is now possible to develop realistic numerical models, through a rigorous procedure of construction, calibration and validation, to simulate the dynamics of these estuarine and coastal ecosystems. These numerical models have now advanced to fully three-dimensional integrated modelling systems composed of coupled circulation, wave, sediment, water quality, light and seagrass models [1, 2]. As the physical and

* Correspondence to: Justin R. Davis, Department of Civil and Coastal Engineering, University of Florida, Gainesville. Florida 32611, U.S.A.
† E-mail: davis@coastal.ufl.edu
‡ E-mail: pete@coastal.ufl.edu

Contract grant/sponsor: National Center for Environmental Research and Quality Control, USEPA
Contract grant/sponsor: University of Florida

Received 25 June 2002
Revised 6 February 2003

ecological processes become better understood, these integrated models are growing larger and larger and hence, so are their computational demands. In addition to the pure complexity of these numerical modelling systems, these models are now being applied to both larger domains and over larger time scales making the computational demands even greater.

One way the increased computational requirements of these estuarine and coastal models has been dealt with is through parallelism. Parallel processing techniques have been applied to numerous hydrodynamic models such as: the Miami Isopycnic Coordinate Ocean Model (MICOM) [3], the CH3D-WES Model [4–6] and the Princeton Ocean Model (POM) [7]. Recently, these techniques have even been applied to the fully coupled integrated modelling system, CH3D-IMS [8, 9]. These studies focused on converting previously serial codes to parallel; however, because of the traditionally serial nature of these legacy models (POM, CH3D, etc.), conversion to a parallel structure usually does not achieve the best efficiency. Hence, it is the goal of this study to develop a parallel model from the ground up with no serial code at all.

The fully parallel model developed herein is designed for a specific purpose: to simulate the storm surge in a large coastal area caused by a major storm (e.g. a hurricane). Hurricanes are among the most dangerous and damaging natural phenomena. As a hurricane approaches and impacts the coastline, storm surge flooding, strong winds, torrential rainfall and tornados can reek havoc on both coastal and inland communities. Of all these impacts, storm surge accounts for over 90% of hurricane related deaths [10]; hence, the increasing interest in simulating such phenomena.

The parallel model developed herein has a two-dimensional formulation and includes a unique combination of features: the ability to handle wetting and drying of computational cells automatically, semi-implicit solution of the propagation, bottom friction and Coriolis terms, a parallel Eulerian–Lagrangian scheme for the solution of the advective and diffusive terms, and a coupled storm model. After the development and verification of the model is presented, an example simulation of storm surge in the Indian River Lagoon area of Florida is presented.

2. GOVERNING DIFFERENTIAL EQUATIONS

The governing three-dimensional Cartesian equations describing free surface flows can be derived from the Navier–Stokes equations. After Reynold's averaging, and applying the hydrostatic approximation, the continuity equation and x- and y-momentum equations for a constant density fluid have the following form [11]:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{1}$$

$$\frac{Du}{Dt} = -g\frac{\partial \zeta}{\partial x} + fv + A_H\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(A_v\frac{\partial u}{\partial z}\right) \tag{2}$$

$$\frac{Dv}{Dt} = -g\frac{\partial \zeta}{\partial y} - fu + A_H\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(A_v\frac{\partial v}{\partial z}\right) \tag{3}$$

where $u(x, y, z, t)$, $v(x, y, z, t)$, $w(x, y, z, t)$ are the velocity components in the horizontal, x- and y-, and the vertical, z-directions; t is the time; $\zeta(x, y, t)$ is the free surface elevation; g is the

gravitational acceleration; A_H and A_v are the horizontal and vertical turbulent eddy coefficients, respectively; and D/Dt is the total derivative, $\partial/\partial t + u\partial/\partial x + v\partial/\partial y + w\partial/\partial z$.

3. VERTICAL BOUNDARY CONDITIONS

The boundary conditions at the free surface ($z = \zeta$) are

$$A_v \frac{\partial u}{\partial z} = \frac{\tau_x^w}{\rho}$$
$$A_v \frac{\partial v}{\partial z} = \frac{\tau_y^w}{\rho} \tag{4}$$

The wind stress is determined from the wind velocity using

$$\tau_x^w = \rho_a C_{ds} u_w \sqrt{u_w^2 + v_w^2}$$
$$\tau_y^w = \rho_a C_{ds} v_w \sqrt{u_w^2 + v_w^2} \tag{5}$$

where τ_x^w and τ_y^w are the components of the wind stress, ρ_a is the density of air (1.293 kg/m^3), u_w and v_w are the components of the wind speed measured at some height about the water surface and C_{ds} is the wind speed drag coefficient.

Many possible formulations for the drag coefficient are available. Table I illustrates several formulations popular with storm surge models. As can be seen in Figure 1, the formulations produce fairly linear curves for wind speeds above 10 m/s, with the exception of the Van Dorn [12] formulation which is significantly less at higher wind speeds. The two cases of the Hsu [13] formulation, which takes into account surface roughness through significant wave

Table I. Descriptions of popular wind stress drag coefficients applied to storm surge models. W_s is the magnitude of wind speed (m/s) measured at 10 m above the water surface, ρ_w and ρ_a are the densities of water and air, respectively, $k_1 = 1.1 \times 10^{-6}$, $k_2 = 2.5 \times 10^{-6}$, $W_{cr} = 7.2$ m/s (14 knots), T_p is the wave period at the spectral peak, H_s is the significant wave height and g is the gravitational acceleration.

Author(s)	Wind stress drag coefficient (C_{ds})
Van Dorn [12]	$\frac{\rho_w}{\rho_a} \times \begin{cases} k_1, & \text{if } W_s < W_{cr} \\ k_1 + k_2 \left(1 - \frac{W_{cr}}{W_s}\right)^2, & \text{if } W_s \geqslant W_{cr} \end{cases}$
Smith and Banke [14]	$0.001 \times (0.63 + 0.066 W_s)$
Garratt [15]	$0.001 \times (0.75 + 0.067 W_s)$
Hsu [13]	$0.16 \times \left[11.0 - \ln\left(\frac{H_s}{(\frac{gT_p}{2\pi W_s})^{2.6}}\right)\right]^{-2}$

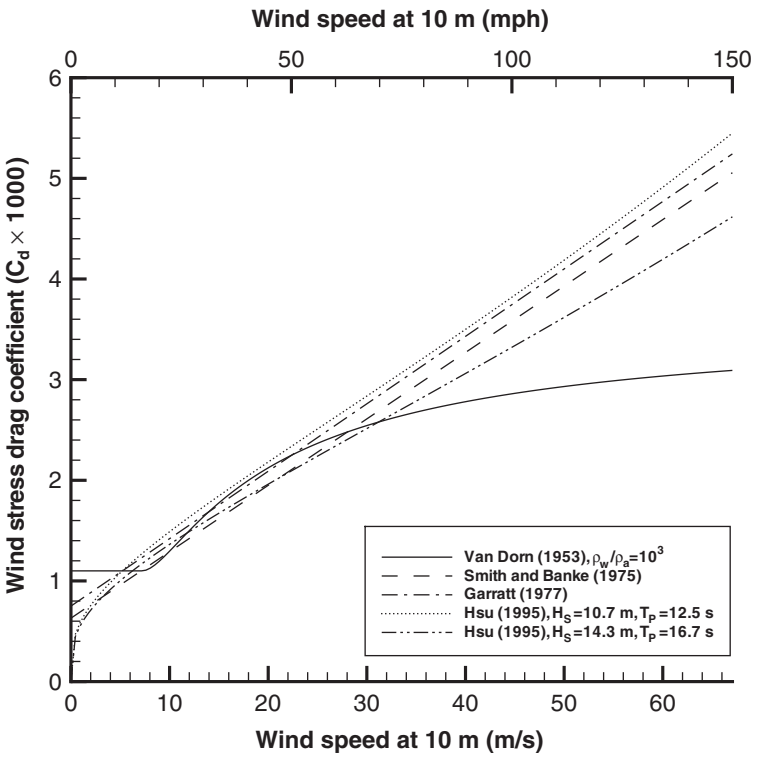


Figure 1. Calculated wind stress drag coefficients using several popular formulations.

height and period, are for Hurricane Kate (1985) in the Gulf of Mexico and Hurricane Gloria (1985) in the Atlantic Ocean, respectively. However, while this formulation is more rigorous, it does not produce markedly different curves than the much simpler linear formulations of Smith and Banke [14] and Garratt [15]. For simplicity, the Garratt [15] formulation is used in the PEM.

The boundary conditions at the bottom ($z = -h$) are

$$A_v \frac{\partial u}{\partial z} = \frac{\tau_x^b}{\rho}$$
$$A_v \frac{\partial v}{\partial z} = \frac{\tau_y^b}{\rho} \tag{6}$$

The bottom stress is given using

$$\tau_x^b = \frac{\rho g U |V|}{C_z^2}$$
$$\tau_y^b = \frac{\rho g V |V|}{C_z^2} \tag{7}$$

where the Chezy coefficient, C_z is given by

$$C_z = 1.0 \times \frac{R^{1/6}}{n} \tag{8}$$

and the hydraulic radius, R is given in meters and n is Manning's n. In shallow estuaries, the hydraulic radius is approximated by the total depth.

4. DIFFERENTIAL EQUATIONS FOR THE MODEL

Vertically averaging the continuity and momentum equations over the depth and applying the vertical boundary conditions yield the following equations for continuity and x- and y-momentum

$$\frac{\partial \zeta}{\partial t} + \frac{\partial}{\partial x}(UH) + \frac{\partial}{\partial y}(VH) = 0 \tag{9}$$

$$\frac{DU}{Dt} = -g\frac{\partial \zeta}{\partial x} + A_H\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) - \frac{C_\mathrm{d} U|V|}{H} + fV + \frac{\tau_x^w}{\rho H} \tag{10}$$

$$\frac{DV}{Dt} = -g\frac{\partial \zeta}{\partial y} + A_H\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right) - \frac{C_\mathrm{d} V|V|}{H} - fU + \frac{\tau_y^w}{\rho H} \tag{11}$$

where H is the total depth $(h+\zeta)$; U and V are the depth averaged velocities $(1/H)\int_{-h}^{\zeta} u\,\mathrm{d}z$ and $(1/H)\int_{-h}^{\zeta} v\,\mathrm{d}z$, respectively; C_d is a dimensional drag coefficient defined as g/C_z^2; and D/Dt is the total derivative, $\partial/\partial t + U\partial/\partial x + V\partial/\partial y$.

5. FINITE DIFFERENCE EQUATIONS FOR THE EXPLICIT MODE

A space-staggered grid system is used to discretize the differential equations [11]. In the staggered grid system, elevation is defined at the centre of a cell, the U-velocity at the left and right sides of a cell, and V-velocity at the top and bottom sides of a cell.

The governing differential equations as shown in Equations (9)–(11) are first given an explicit discretization of the propagation and bottom friction terms:

$$\frac{\zeta_{i,j}^{n+1} - \zeta_{i,j}^{n}}{\Delta t} + \frac{(U_{i+1,j}^{n} H_{u,i+1,j}^{n} - U_{i,j}^{n} H_{u,i,j}^{n})}{\Delta x} + \frac{(V_{i,j+1}^{n} H_{v,i,j+1}^{n} - V_{i,j}^{n} H_{v,i,j}^{n})}{\Delta y} = 0 \tag{12}$$

$$\frac{U_{i,j}^{n+1} - U_{i,j}^{n}}{\Delta t} = -\frac{g}{\Delta x}(\zeta_{i,j}^{n} - \zeta_{i-1,j}^{n}) - \frac{C_\mathrm{d}}{H_{u,i,j}^{n}}|V|_{u,i,j}^{n} U_{i,j}^{n} + \frac{\tau_{x,i,j}^{w}}{\rho H_{u,i,j}^{n}} + F_{u,i,j}^{n} \tag{13}$$

$$\frac{V_{i,j}^{n+1} - V_{i,j}^{n}}{\Delta t} = -\frac{g}{\Delta y}(\zeta_{i,j}^{n} - \zeta_{i,j-1}^{n}) - \frac{C_\mathrm{d}}{H_{v,i,j}^{n}}|V|_{v,i,j}^{n} V_{i,j}^{n} + \frac{\tau_{y,i,j}^{w}}{\rho H_{v,i,j}^{n}} + F_{v,i,j}^{n} \tag{14}$$

where the subscripts u and v represent a quantity at either the u-node or v-node in the staggered grid system, H represents the total depth, and F represents the remaining non-linear, diffusion

and pressure gradient terms. The staggered schemes presented in Equations (12)–(14) possess second-order spatial accuracy.

6. FINITE DIFFERENCE EQUATIONS FOR THE SEMI-IMPLICIT MODE

The governing differential equations as shown in Equations (9)–(11) are given a semi-implicit discretization of the propagation and bottom friction terms, with the Coriolis term given a semi-implicit discretization only in the y-momentum equation:

$$\begin{aligned}&\frac{\zeta_{i,j}^{n+1}-\zeta_{i,j}^{n}}{\Delta t}+\frac{\theta_1}{\Delta x}(U_{i+1,j}^{n+1}H_{u,i+1,j}^{n}-U_{i,j}^{n+1}H_{u,i,j}^{n})\\&+\frac{\theta_1}{\Delta y}(V_{i,j+1}^{n+1}H_{v,i,j+1}^{n}-V_{i,j}^{n+1}H_{v,i,j}^{n})\\&+\frac{1-\theta_1}{\Delta x}(U_{i+1,j}^{n}H_{u,i+1,j}^{n}-U_{i,j}^{n}H_{u,i,j}^{n})\\&+\frac{1-\theta_1}{\Delta y}(V_{i,j+1}^{n}H_{v,i,j+1}^{n}-V_{i,j}^{n}H_{v,i,j}^{n})=0\end{aligned}\tag{15}$$

$$\begin{aligned}\frac{U_{i,j}^{n+1}-U_{i,j}^{n}}{\Delta t}=&-\frac{g\theta_1}{\Delta x}(\zeta_{i,j}^{n+1}-\zeta_{i-1,j}^{n+1})-\frac{g(1-\theta_1)}{\Delta x}(\zeta_{i,j}^{n}-\zeta_{i-1,j}^{n})\\&-\frac{C_{\mathrm{d}}}{H_{u,i,j}^{n}}|V|_{u,i,j}^{n}U_{i,j}^{n+1}\theta_2-\frac{C_{\mathrm{d}}}{H_{u,i,j}^{n}}|V|_{u,i,j}^{n}U_{i,j}^{n}(1-\theta_2)\\&+fV_{u,i,j}^{n}+\frac{\tau_{x,i,j}^{w}}{\rho H_{u,i,j}^{n}}+F_{u,i,j}^{n}\end{aligned}\tag{16}$$

$$\begin{aligned}\frac{V_{i,j}^{n+1}-V_{i,j}^{n}}{\Delta t}=&-\frac{g\theta_1}{\Delta y}(\zeta_{i,j}^{n+1}-\zeta_{i,j-1}^{n+1})-\frac{g(1-\theta_1)}{\Delta y}(\zeta_{i,j}^{n}-\zeta_{i,j-1}^{n})\\&-\frac{C_{\mathrm{d}}}{H_{v,i,j}^{n}}|V|_{v,i,j}^{n}V_{i,j}^{n+1}\theta_2-\frac{C_{\mathrm{d}}}{H_{v,i,j}^{n}}|V|_{v,i,j}^{n}V_{i,j}^{n}(1-\theta_2)\\&-fU_{v,i,j}^{n+1}\theta_3-fU_{v,i,j}^{n}(1-\theta_3)+\frac{\tau_{y,i,j}^{w}}{\rho H_{v,i,j}^{n}}+F_{v,i,j}^{n}\end{aligned}\tag{17}$$

where $F_{u,i,j}^{n}$ and $F_{v,i,j}^{n}$ represent the remaining non-linear, diffusion and pressure gradient terms in the x- and y-directions, respectively and θ_1, θ_2, and θ_3 are the degrees of implicitness of the surface slope, bottom friction and Coriolis terms, respectively. The staggered schemes presented in Equations (15)–(17) possess second-order spatial accuracy.

The x-momentum equation (16) is first solved for $U_{i,j}^{n+1}$ and then substituted into the Coriolis term in the y-momentum equation (17) using the following equation

$$-fU_{v,i,j}^{n+1}\theta_3=-f\,\frac{(U_{i,j}^{n+1}+U_{i+1,j}^{n+1}+U_{i,j-1}^{n+1}+U_{i+1,j-1}^{n+1})}{4}\theta_3\tag{18}$$

The new y-momentum finite difference equation is then solved for $V_{i,j}^{n+1}$. The i's in the $U_{i,j}^{n+1}$ equation and the j's in the $V_{i,j}^{n+1}$ are incremented by one yielding $U_{i+1,j}^{n+1}$ and $V_{i,j+1}^{n+1}$, respectively. The resulting 4 momentum finite difference equations, $U_{i,j}^{n+1}$, $U_{i+1,j}^{n+1}$, $V_{i,j}^{n+1}$, and $V_{i,j+1}^{n+1}$, are substituted back into the continuity equation (15) resulting in the following 9-diagonal system of linear equations for the surface elevation, $\zeta_{i,j}^{n+1}$

$$\begin{aligned}
&\Pi_{nw}\zeta_{i-1,j+1}^{n+1} + \Pi_{n}\zeta_{i,j+1}^{n+1} + \Pi_{ne}\zeta_{i+1,j+1}^{n+1} \\
&\quad + \Pi_{w}\zeta_{i-1,j}^{n+1} + \Pi_{c}\zeta_{i,j}^{n+1} + \Pi_{e}\zeta_{i+1,j}^{n+1} \qquad = (RHS)_{i,j}^{n} \\
&\quad + \Pi_{sw}\zeta_{i-1,j-1}^{n+1} + \Pi_{s}\zeta_{i,j-1}^{n+1} + \Pi_{se}\zeta_{i+1,j-1}^{n+1}
\end{aligned} \tag{19}$$

where

$$\begin{aligned}
\Pi_{nw} &= -\frac{\Gamma_{i,j+1}}{4}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i,j+1} \\
\Pi_{ne} &= +\frac{\Gamma_{i,j+1}}{4}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i+1,j+1} \\
\Pi_{n} &= -\Pi_{nw} - \Pi_{ne} - \Phi_{v,i,j+1} \\
\Pi_{sw} &= +\frac{\Gamma_{i,j}}{4}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i,j-1} \\
\Pi_{se} &= -\frac{\Gamma_{i,j}}{4}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i+1,j-1} \\
\Pi_{s} &= -\Pi_{sw} - \Pi_{se} - \Phi_{v,i,j} \\
\Pi_{w} &= +\frac{\Gamma_{i,j}}{4}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i,j} - \frac{\Gamma_{i,j+1}}{4}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i,j} - \Phi_{u,i,j} \\
\Pi_{e} &= -\frac{\Gamma_{i,j}}{4}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i+1,j} + \frac{\Gamma_{i,j+1}}{4}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i+1,j} - \Phi_{u,i+1,j} \\
\Pi_{c} &= 1 - \Pi_{w} - \Pi_{e} - \Pi_{n} - \Pi_{nw} - \Pi_{ne} - \Pi_{s} - \Pi_{sw} - \Pi_{se}
\end{aligned} \tag{20}$$

$$\begin{aligned}
(RHS)_{i,j}^{n} = \zeta_{i,j}^{n} &+ \frac{\Delta t}{\Delta x}\theta_1 H_{u,i,j}^{n}\alpha_{u,i,j}\beta_{u,i,j} \\
&- \frac{\Delta t}{\Delta x}\theta_1 H_{u,i+1,j}^{n}\alpha_{u,i+1,j}\beta_{u,i+1,j} \\
&+ \frac{\Delta t}{\Delta y}\theta_1 H_{v,i,j}^{n}\alpha_{v,i,j}\beta_{v,i,j} \\
&- \frac{\Delta t}{\Delta y}\theta_1 H_{v,i,j+1}^{n}\alpha_{v,i,j+1}\beta_{v,i,j+1} \\
&+ \frac{\Delta t(1-\theta_1)}{\Delta x}(U_{i,j}^{n}H_{u,i,j}^{n} - U_{i+1,j}^{n}H_{u,i+1,j}^{n})
\end{aligned}$$

$$+\frac{\Delta t(1-\theta_1)}{\Delta y}(V^n_{i,j}H^n_{v,i,j}-V^n_{i,j+1}H^n_{v,i,j+1})$$
$$-\frac{\Gamma_{i,j}}{4}\begin{pmatrix}\alpha_{u,i,j}\beta_{u,i,j}+\alpha_{u,i+1,j}\beta_{u,i+1,j}\\+\alpha_{u,i,j-1}\beta_{u,i,j-1}+\alpha_{u,i+1,j-1}\beta_{u,i+1,j-1}\end{pmatrix}$$
$$+\frac{\Gamma_{i,j+1}}{4}\begin{pmatrix}\alpha_{u,i,j+1}\beta_{u,i,j+1}+\alpha_{u,i+1,j+1}\beta_{u,i+1,j+1}\\+\alpha_{u,i,j}\beta_{u,i,j}+\alpha_{u,i+1,j}\beta_{u,i+1,j}\end{pmatrix} \tag{21}$$

$$\alpha_{u,i,j}=\Delta tF^n_{u,i,j}+U^n_{i,j}+fV^n_{u,i,j}\Delta t-\frac{g\Delta t}{\Delta x}(\zeta^n_{i,j}-\zeta^n_{i-1,j})(1-\theta_1)$$
$$-\frac{C_{\mathrm{d}}\Delta t}{H^n_{u,i,j}}|V|^n_{u,i,j}U^n_{i,j}(1-\theta_2)+\frac{\Delta t\tau^w_{x,i,j}}{\rho H^n_{u,i,j}}+\Delta tF^n_{u,i,j} \tag{22}$$

$$\alpha_{v,i,j}=\Delta tF^n_{v,i,j}+V^n_{i,j}$$
$$-fU^n_{v,i,j}\Delta t(1-\theta_3)$$
$$-\frac{g\Delta t}{\Delta y}(\zeta^n_{i,j}-\zeta^n_{i,j-1})(1-\theta_1)$$
$$-\frac{C_{\mathrm{d}}\Delta t}{H^n_{v,i,j}}|V|^n_{v,i,j}V^n_{i,j}(1-\theta_2)+\frac{\Delta t\tau^w_{y,i,j}}{\rho H^n_{v,i,j}}+\Delta tF^n_{v,i,j} \tag{23}$$

$$\beta_{u,i,j}=\frac{1}{1+\frac{C_{\mathrm{d}}\Delta t}{H^n_{u,i,j}}|V|^n_{u,i,j}\theta_2} \tag{24}$$

$$\beta_{v,i,j}=\frac{1}{1+\frac{C_{\mathrm{d}}\Delta t}{H^n_{v,i,j}}|V|^n_{v,i,j}\theta_2} \tag{25}$$

$$\Gamma_{i,j}=f\theta_1\theta_3\frac{\Delta t}{\Delta y}H^n_{v,i,j}\beta_{v,i,j}\Delta t \tag{26}$$

$$\Phi_{u,i,j}=\frac{\Delta t}{\Delta x}\theta_1H^n_{u,i,j}\frac{g\Delta t}{\Delta x}\theta_1\beta_{u,i,j} \tag{27}$$

$$\Phi_{v,i,j}=\frac{\Delta t}{\Delta y}\theta_1H^n_{v,i,j}\frac{g\Delta t}{\Delta y}\theta_1\beta_{v,i,j} \tag{28}$$

7. THE PARALLEL EULERIAN–LAGRANGIAN METHOD (ELM)

A major difficulty in numerical modelling of free surface flows is the accurate and efficient treatment of the advection and diffusion terms in the governing equations. Numerous numerical schemes are available to solve these terms; however, few are unconditionally stable and most have stringent time step limitations. Since the intended use of the PEM is in large scale

applications with a large number of grid cells, it is important that the model uses time steps as large as possible to minimize simulation run time. Hence, the optimal method for the PEM should be unconditionally stable. To this end, the Eulerian–Lagrangian method (ELM) is used in the PEM to solve the advection and diffusion terms in the governing equations.

The ELM uses the Lagrangian form of the governing equations in an Eulerian computational grid system. The method is generally referred to as the semi-Lagrangian method in the numerical weather prediction community [16], but is called the Eulerian–Lagrangian method by estuarine and coastal modelers (e.g. References [17, 18]). Previous studies have also looked at the challenge of developing parallel ELM algorithm. Behrens [19] developed a parallel semi-Lagrangian advective scheme for use with an adaptive grid system; however, it was designed for used on a KSR-1 (Kendall Square Research), a virtual shared memory computer, and would not work well on a distributed memory system. Malevsky and Thomas [20] also discussed parallel ELM algorithms; however, their methods were based sharing boundary data between processors.

Following Casulli and Cheng [17], the portion of F_u representing the remaining advection and diffusion terms in the x-direction can be discretized using the ELM method resulting in (a similar equation can be developed for F_v, the remaining advection and diffusion terms in the y-direction)

$$F^n_{u,i,j} = u^n_{i-a,j-b} + A_H \Delta t \left(\frac{u^n_{i-a-1,j-b} - 2u^n_{i-a,j-b} + u^n_{i-a+1,j-b}}{\Delta x^2} \right) + A_H \Delta t \left(\frac{u^n_{i-a,j-b-1} - 2u^n_{i-a,j-b} + u^n_{i-a,j-b+1}}{\Delta y^2} \right) \tag{29}$$

where $a = u\Delta t/\Delta x$ and $b = v\Delta t/\Delta y$ are the grid Courant numbers.

Because of the non-linearity of the advective terms, the determination of a and b requires the integration of the streak lines, $\mathrm{d}x/\mathrm{d}t = u$ and $\mathrm{d}y/\mathrm{d}t = v$, in which case the right-hand sides are known only at time level n. Thus, u and v are assumed to be invariant over a time step and the streak lines will be integrated numerically backwards from time level $n+1$ to n using the Euler method, since the streak lines, which in general are not straight lines, are better approximated.

In general, a and b are not integers; therefore $(i - a, j - b)$ is not a grid point and an interpolation formula must be used to solve for $u^n_{i-a,j-b}$. For positive a and b, let l and m be the integer parts and p and q be their corresponding decimal parts, so that $a = l + p$ and $b = m + q$. Then $u^n_{i-a,j-b}$ can be approximated as

$$u^n_{i-a,j-b} = (1-p)[(1-q)u^n_{i-l,j-m} + qu^n_{i-l,j-m-1}] + p[(1-q)u^n_{i-l-1,j-m} + qu^n_{i-l-1,j-m-1}] \tag{30}$$

In the PEM, each processor controls only one small portion of the overall grid system. For each u and v node on a given processor, a streak line is tracked backwards. Depending on the direction of the flow and the proximity of the node to a processor boundary, the streak lines may be tracked to another processor. When this occurs, the original processor which started

back-tracking the streak line does not have enough information to proceed. To handle these type of situations a parallel ELM algorithm was developed.

The parallel ELM algorithm consists of two parts. The first part attempts to trace back every needed streak line within a processors domain (Listing 1). If the streak line is tracked to another processor, a message is sent to the appropriate neighbour processor giving it enough information about the current progress of the back-tracked streak line such that the neighbour processor can finish the backward integration. Once this message has been sent, a counter identifying the number of uncompleted streak lines is incremented (NUM_OUTSIDE_GRID) and the processor continues to track backwards the remaining nodes streak lines. Once all streak lines have been tracked backwards or messages sent to neighbour processors telling them to complete the streak line, the first part of the algorithm is complete.

In the second part of the algorithm (Listing 2), each processor spins in a loop which (1) checks for completed streak lines sent back from other processors (Lines 63–68), (2) checks to see if any neighbour processors have requested that they complete one of their streak lines (Lines 77–101), and (3) check to see if all of its unfinished streak lines have been returned (Lines 107–113). As completed streak lines are returned from neighbour processors, a counter identifying the number of returned completed streak lines is incremented

Listing 1. Pseudocode representation of the parallel ELM algorithm (Part 1/2).

```
1  ! Backtrack (as far as possible) all of the streak lines
2  ! which start in the local subdomain.
3    do J=LOCAL_J_MIN, LOCAL_J_MAX          ! Local subdomain range of J
4      do I=LOCAL_I_MIN, LOCAL_I_MAX        ! Local subdomain range of I
5
6        { Initialize the streak line position.}
7        COUNT=0                            ! Integration steps
8        OTHER_SUB_GRID_FLAG=0              ! Location of streak line
9        NUM_OUTSIDE_GRID=0                 ! Num. of streak lines outside local grid
10
11 !     Track the streak line backwards a total of NITER steps or if the position of
12 !     the line goes outside the local grid, send the streak line's position to the
13 !     processor containing that section of the grid.
14       do while ( COUNT < NITER .and. OTHER_SUB_GRID_FLAG == 0 )
15         COUNT++                          ! Increment integration step
16         {Update the streak line position by interpolation of the
17          velocity field at the previous position.}
18         if ( {updated streak line position is outside the local grid} ) then
19           OTHER_SUB_GRID_FLAG=1          ! Set flag to exit integration loop
20           NUM_OUTSIDE_GRID++             ! Increment number of streak lines
21                                          !   outside the local grid
22           {Send a non-blocking message to the processor which contains the streak line
23            at its updated position. This message contains the updated streak line
24            position, its starting position, the starting processor number and the
25            current integration step}
26         endif
27
28 ! If the backward integration is complete, calculate the variable at
29 ! its final streak position.
30         if ( COUNT == NITER .and. OTHER_SUB_GRID_FLAG == 0 ) then
31           {Calculate the variable at its final streak position.}
32         endif
33
34       end do                             ! do while ...
35
36     end do                               ! do I=...
37   end do                                 ! do J=...
```


Listing 2. Pseudocode representation of the parallel ELM algorithm (Part 2/2).

```
38  ! Inform other processors when this process has received back all
39  ! the streak lines it sent out
40    INFORM_OTHERS=1
41  ! Number of streak lines sent out that this processor has received back
42    NUM_MESSAGES_RECEIVED=0
43    COMPLETED=.false.                          ! Status of all the processors
44
45  ! Loop continuously until all processors have received
46  !   all of their streak lines back
47    do while ( COMPLETED == .false. )
48      {Non-blocking check for incoming streak lines.}
49      if ( {Incoming streak line exists} ) then
50
51        {Accept the incoming streak line.}
52
53        if ( { Integration step of incoming streak line == NITER} ) then
54
55          SENT_ON=0                            ! Flag to indicate if
56                                               !   a message was sent
57          if ( {Message final destination is not this processor} ) then
58            {Perform one last backward integration}
59            {Send a non-blocking message to the processor which contains the streak
60             line at its updated position. This message contains the updated streak
61             line position, its starting position, the starting processor number and
62             the current integration step.}
63            SENT_ON=1
64          endif
65
66  !       If message was not sent on, its final destination is this processor
67          if ( SENT_ON /= 1) then              ! Message was not sent on
68            {Save the variable information passed in the message.}
69            NUM_MESSAGES_RECEIVED++            ! Increment the count of messages
70                                               !   received whose final destination
71                                               !   is this processor
72          endif
73
74        else                                   ! Backtrack to get variable
75
76          { Initialize the streak line position to that of the message.}
77  !       Track the streak line backwards a total of NITER steps or if the position
78  !       of the streak line goes outside the local grid, send the streak line's
79  !       position to the processor containing that section of the grid.
80          do while ( COUNT < NITER .and. OTHER_SUB_GRID_FLAG == 0 )
81            COUNT++                            ! Increment integration step
82            {Update the streak line position by interpolation of the
83              velocity field at the previous position.}
84            if ( {updated streak line position is outside the local grid} ) then
85              OTHER_SUB_GRID_FLAG=1            ! Set flag to exit integration loop
86              {Send a non-blocking message to the processor which contains the streak
87               line at its updated position. This message contains the updated streak
88               line position, its starting position, the starting processor number and
89               the current integration step.}
90            endif
91          end do                               ! do while ...
92
93  !       If the backward integration is complete, calculate the variable at its final
94  !       streak position and send the result to its processor of origin.
95          if ( COUNT == NITER .and. OTHER_SUB_GRID_FLAG == 0 ) then
96            {Calculate the variable at its final streak position.}
97            {Send a non-blocking message to the processor which contains the streak line
98             at its updated position. This message contains the updated streak line
99             position, its starting position, the starting processor number and the
100            current integration step.}
101         endif
102
```



```
103       endif                                    ! Integration step == NITER
104
105     endif                                      ! Incoming streak line
106
107   !  If this processor has received back all of the streak lines it sent to other
108   !  processors, then send a signal to the other processors.
109     if ( NUM_MESSAGES_RECEIVED == NUM_OUTSIDE_GRID .and.
110          INFORM_OTHERS == 1 ) then
111       {Send completion signal to all other processors.}
112       INFORM_OTHERS=0                          ! Only inform others once
113     endif
114
115   !  Check for completion signal from other processors
116     {Non-blocking check for completion signal.}
117     if ( {A completion signal is present} ) then
118       {Read and record which processor sent the signal.}
119     endif
120
121   !  If this processor has received all of its streak lines then check to see if all of
122   !  the other processors have completed their streak lines, if so then terminate the
123   !  main loop.
124     if ( {This processor has received all its streak lines back} ) then
125       if ( {All other processors have received their streaks lines back} ) then
126         COMPLETED=.true.                       ! Flag will terminate main loop
127       endif
128     endif
129
130   end do                                       ! do while ...
```

(NUM_MESSAGES_RECEIVED). Once all of a given processors unfinished streak lines have been returned (NUM_MESSAGES_RECEIVED = NUM_OUTSIDE_GRID), a message is sent to all other processors telling them that this processor has completed (Line 111). Upon receipt of a similar message from all other processors, the processors then exit their respective spin loops (Lines 115–128).

8. STORM MODEL EQUATIONS

One of the most important parts of a numerical model used to simulate storm surge is the storm model itself. A good storm model is necessary to determine rapidly changing atmospheric pressure gradient and wind stress associated with the passage of a storm. While a planetary boundary layer (PBL) model [21–23] would be the best choice for a storm model, a PBL model is too complex for the purposes of this study. Instead a simple storm model assuming an exponential decay of pressure from the center of a storm is used. The governing pressure gradient terms and wind field (used to calculate the wind stresses, τ_x and τ_y) can be derived as follows.

The local air pressure in a storm, P, can be written as [24]

$$P = P_o + (P_\infty - P_o)e^{-A/r^B} \tag{31}$$

where P_o is the pressure in the centre on the storm, P_∞ is the freestream pressure, r is the distance from the centre of the storm and A and B are scaling parameters. However, for the purposes of this study, the simpler local air pressure formulation of Wilson [25] is used (A is

set equal to the radius of maximum wind speed, R, and B is set equal to 1). The remaining derivation then follows in the manner of Wilson [25].

Subtracting P_∞ from both sides and rearranging the right-hand side yields,

$$P - P_\infty = (P_o - P_\infty)(1 - e^{-R/r}) \tag{32}$$

The left-hand side is the relative atmospheric pressure, P_a, and $P_o - P_\infty$ is the central pressure drop of the storm, ΔP_o. Making these substitutions, the equation can be rewritten as

$$P_a = \Delta P_o(1 - e^{-R/r}) \tag{33}$$

The air pressure term in the x- and y-momentum equations consists of derivatives of P_a with respect to x and y, respectively. The terms, after negating and dividing by water density are

$$-\frac{1}{\rho_w}\frac{\partial P_a}{\partial x} = \frac{\Delta P_o}{\rho_w}\frac{\partial}{\partial x}(e^{-R/r}) \tag{34}$$

$$-\frac{1}{\rho_w}\frac{\partial P_a}{\partial y} = \frac{\Delta P_o}{\rho_w}\frac{\partial}{\partial y}(e^{-R/r}) \tag{35}$$

The finite difference forms of the x- and y-momentum air pressure terms used in the model are evaluated at the u- and v-nodes, respectively

$$-\frac{1}{\rho_w}\frac{\partial P_a}{\partial x} = \frac{\Delta P_o}{\rho_w \Delta x}(e^{-R/r_{i,j}} - e^{-R/r_{i-1,j}}) \tag{36}$$

$$-\frac{1}{\rho_w}\frac{\partial P_a}{\partial y} = \frac{\Delta P_o}{\rho_w \Delta y}(e^{-R/r_{i,j}} - e^{-R/r_{i,j-1}}) \tag{37}$$

The distance, $r_{i,j}$, is measured from the centre of the storm to the center of the (i,j) cell.

The storm also influences the elevation at open boundaries, $\zeta_{open} = \zeta_{tide} + \zeta_{storm}$. The surface elevation due to the pressure of the storm can be written as

$$\zeta_{storm} = \frac{P_a}{\rho_w g} = \frac{\Delta P_o}{\rho_w g}(1 - e^{-R/r}) \tag{38}$$

The cyclostrophic wind velocity, U_c, is

$$U_c = \sqrt{-\frac{\Delta P_o}{\rho_a}\frac{R}{r}e^{-R/r}} \tag{39}$$

The geostrophic wind velocity, U_g, is

$$U_g = -\frac{\Delta P_o}{f\rho_a}\frac{R}{r^2}e^{-R/r} \tag{40}$$

The gradient wind velocity, U_G is

$$U_G = U_c(\sqrt{\gamma^2 + 1} - \gamma) \tag{41}$$

where

$$\gamma = \frac{1}{2}\left(\frac{V_s^*}{U_c} + \frac{U_c}{U_g}\right) \tag{42}$$

and the resolved part, V_s^*, of the translational velocity of the storm, V_s is

$$V_s^* = V_s \sin(\theta) \tag{43}$$

where θ is the angle from the direction of bearing of the storm, β, to any point inside the storm. The surface wind velocity, U_s, in the x- and y-directions is then written as

$$U_{sx} = KU_G \cos(90 + \theta + \beta + \phi) \tag{44}$$

$$U_{sy} = KU_G \sin(90 + \theta + \beta + \phi) \tag{45}$$

where ϕ is an inward rotation angle of 18° and K is the ratio of surface wind velocity to gradient wind velocity.

9. SOLUTION TECHNIQUE

When performing simulations, PEM operates in either an explicit or semi-implicit mode. Explanation of how the water level and velocities are calculated in these modes follows. The message passing technique that is used to explicitly pass data between processors conforms to the message passing interface (MPI) standard. Specifically, MPICH [26], a portable version of MPI, libraries are used in the PEM.

9.1. The explicit mode

Due to stringent time step limitations, the explicit mode is generally used only for testing purposes. First, tidal, wind and pressure boundary conditions are updated for the new time level. Then external mode equations are solved on each processor (Equations (12) through (14)). Next, the boundary water levels and velocities which may be used by a given cell's neighbour processors are passed in the order shown in Figure 2. Then, the parallel ELM is used to calculate the non-linear and diffusion terms at the new time level. Finally, the time step is incremented and process of solving the equations begins again.

9.2. The semi-implicit mode

As with the explicit mode, the first step is to update the tidal, wind and pressure gradient boundary conditions for the new time level. Then, the 9-diagonal system of linear equations for water level (Equation (19)) are solved using Aztec (Version 2.1) [27]. Aztec is a parallel iterative library that solves linear systems of equations of the form $Ax = b$, where A is a given $n \times n$ sparse matrix, b is a given vector of length n, and x is the vector of length n to be calculated. Parallel communication in the Aztec routines is based on the MPI standard.

Aztec includes a number of solution and scaling algorithms, preconditioners and residual expressions for determination of convergence; however, not all options are applicable to the

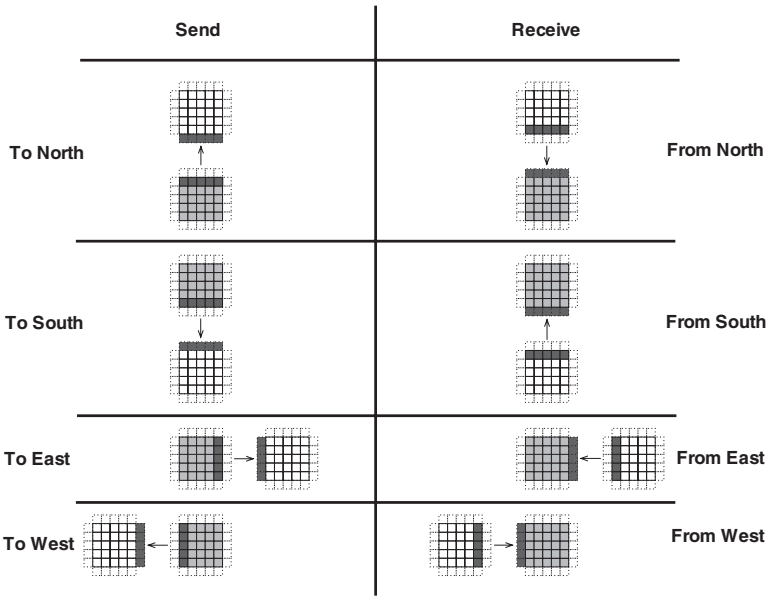


Figure 2. The order of the send and receive operations in the PEM. The cells with dotted lines on the receiving processors are termed 'ghost cells'.

9-diagonal system of equations given in (19). The solution algorithms used in this study are conjugate gradient squared (CGS), transpose-free quasi-minimal residual (TFQMR) and bi-conjugate gradient with stabilization (BICGSTAB). The scaling algorithms used are point Jacobi, scaling each row so the magnitude of its elements sum to 1, symmetric scaling so diagonal elements are 1 and symmetric scaling using the matrix row sums. The residual expression chosen to check for convergence is

$$\frac{\|r\|_2}{\|r^{(0)}\|_2} < \text{Tolerance} \tag{46}$$

Aztec works with two specific sparse matrix formats: (1) a point-entry modified sparse row (MSR) format [28] or (2) a block-entry variable block row (VBR) format [29]. Aztec generalizes these formats for parallel implementation and are referred to as 'distributed' yielding DMSR and DVBR, respectively. Further details on Aztec's implementation of these formats can be found in Tuminaro *et al.* [27]. The PEM uses the MSR format.

After the water level is updated, the velocities are calculated using Equations (16) and (17). Next, as with the external mode, after a given time step has been completed, the boundary water levels and velocities of each processor are passed to its neighbour processors (Figure 2). Then, the parallel ELM is used to calculate the non-linear and diffusion terms at the new time level. Finally, the time step is incremented and the process of solving the equations begins again.

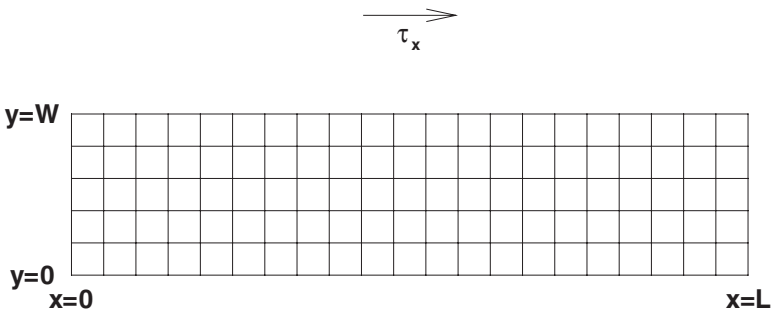


Figure 3. Computational grid for the simple wind forcing analytic test.

Table II. A comparison between the analytic and simulated steady state setup of water level.

x (km)	$\zeta_{\text{theoretical}}$ (cm)	$\zeta_{\text{simulated}}$ (cm)
0.5	−2.04	−2.04
10.5	0	0.00
20.5	+2.04	+2.04

10. MODEL VERIFICATION

Before a numerical model can be applied to a real system, it is necessary to verify the accuracy of the various numerical schemes of the model. This section presents the numerical accuracy of the 2D, parallel, wetting and drying numerical model, PEM, by comparing model results with a number of analytic solutions describing the circulation in idealized basins forced by wind and tide with and without Coriolis acceleration and wetting and drying. All simulations were first performed serially and then in parallel to ensure the accuracy of the parallel algorithms. In addition, the parallel speedup of the explicit scheme and the semi-implicit and implicit schemes are presented along with a discussion of how the characteristics of the iterative solver affect the implicit calculations.

10.1. Wind forcing

The analytical setup due to a constant wind stress in a rectangular basin can be written as

$$\zeta(x) = \frac{\tau_w}{\rho g h}\left(x - \frac{L}{2}\right) \tag{47}$$

where ζ is the setup of the water surface, τ_w is the applied wind stress, H $(=h+\zeta)$ is the total depth of the water column, L is the length of the basin and x is the distance from the left edge. The depth is chosen such that $h \gg \zeta$ so that H can be approximated by h.

The computational grid used in the wind stress test is a 21×5 cell, orthogonal grid with a length, L, of 21 km and a width, W, of 5 km (Figure 3). The depth is a constant 5 m and the grid spacings, Δx and Δy are fixed at 1 km. A constant wind stress of 1 dyne/cm^2 is applied in the positive x-direction and a 900 s time step is used in the model. Table II shows

the comparison between the simulated water level and the water level determined from the analytic solution. The model simulates this analytic test case exactly.

10.2. Tidal forcing with Coriolis

The linearized 2D shallow water equations of continuity, x- and y-momentum without friction or diffusion take the following form:

$$\frac{\partial \eta}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0 \tag{48}$$

$$\frac{\partial U}{\partial t} + c^2 \frac{\partial \eta}{\partial x} - \Omega V = 0 \tag{49}$$

$$\frac{\partial V}{\partial t} + c^2 \frac{\partial \eta}{\partial y} + \Omega U = 0 \tag{50}$$

where η is the surface elevation, U and V are the vertically integrated velocities in the x- and y-directions, respectively, t is time, Ω is the Coriolis parameter and $c = \sqrt{gH}$ is the wave celerity and assuming that the depth, H is much larger than the surface elevation.

Assuming only one tidal constituent with a period $T = 2\pi/\sigma$, these equations can be solved for U, V, and η [30] and are given as

$$\begin{aligned} U(x,y,t) = {} & \frac{\mathrm{i}c^2\sigma\pi}{(\sigma^2-\Omega^2)x_1} \sum_{m=1}^{\infty} mH_m \left[1 + \frac{\Omega^2 k_{2m} x_1^2}{\sigma^2 m^2 \pi^2}\right] \\ & \times \sin\left(\frac{m\pi x}{x_1}\right) \exp(-\mathrm{i}k_{2m}(y_1 - y) - \mathrm{i}\sigma t) \end{aligned} \tag{51}$$

$$\begin{aligned} V(x,y,t) = {} & \frac{H_o m_2 c^2}{\sigma} \left[R \exp\left(\frac{\Omega m_2 x}{\sigma} - \mathrm{i}m_2(y_1 - y)\right)\right] \exp(-\mathrm{i}\sigma t) \\ & - \frac{H_o m_2 c^2}{\sigma} \left[\exp\left(\frac{\Omega m_2 (x_1 - x)}{\sigma} + \mathrm{i}m_2(y_1 - y)\right)\right] \exp(-\mathrm{i}\sigma t) \\ & + \sum_{m=1}^{\infty} H_m \left[\frac{k_{2m}}{\sigma} \cos\left(\frac{m\pi x}{x_1}\right) + \frac{\Omega x_1}{m\pi c^2} \sin\left(\frac{m\pi x}{x_1}\right)\right] \\ & \times \exp(-\mathrm{i}k_{2m}(y_1 - y) - \mathrm{i}\sigma t) \end{aligned} \tag{52}$$

$$\begin{aligned} \eta(x,y,t) = {} & H_o \exp\left(\frac{\Omega m_2 (x_1 - x)}{\sigma} + \mathrm{i}m_2(y_1 - y) - \mathrm{i}\sigma t\right) \\ & + RH_o \exp\left(\frac{\Omega m_2 x}{\sigma} - \mathrm{i}\, m_2(y_1 - y) - \mathrm{i}\sigma t\right) \\ & + \sum_{m=1}^{\infty} H_m \left[\cos\left(\frac{m\pi x}{x_1}\right) + \frac{\Omega k_{2m} x_1}{\sigma m \pi} \sin\left(\frac{m\pi x}{x_1}\right)\right] \\ & \times \exp(-\mathrm{i}k_{2m}(y_1 - y) - \mathrm{i}\sigma t) \end{aligned} \tag{53}$$

where x_1 is the length of the basin in the x-direction, y_1 is the length of the basin in the y-direction and H_o is the amplitude of the tidal forcing at the open boundary. The wave numbers of the mth Poincare mode and that of the Kelvin are given by

$$k_{2m}^2 = -\left[\frac{m^2\pi^2}{x_1^2} - \frac{\sigma^2 - \Omega^2}{c^2}\right] \tag{54}$$

$$m_2^2 = \frac{\sigma^2}{c^2} \tag{55}$$

The unknown constants, R, H_1, H_2, ..., H_N are obtained when the equation for $V(x, y, t) = 0$ is satisfied yielding

$$H_o m_2 \left[R \exp\left(\frac{\Omega m_2 x}{\sigma}\right) - \exp\left(\frac{\Omega m_2 (x_1 - x)}{\sigma}\right)\right] + \sum_{m=1}^{\infty} H_m \left[k_{2m} \cos\left(\frac{m\pi x}{x_1}\right) + \frac{\Omega x_1 \sigma}{m\pi c^2} \sin\left(\frac{m\pi x}{x_1}\right)\right] = 0 \tag{56}$$

To solve for the unknown constants, the equation is truncated at the Nth term leaving $N + 1$ unknowns. The equation is then applied to $N + 1$ points on $(0, x_1)$ resulting in $N + 1$ simultaneous equations for the $N + 1$ unknowns which are then solved. As N is increased, a converging sequence of values is found for R and each of the H_m's. The computational grid used in the wind stress test is a 40×40 cell, orthogonal grid with a length and width of 41 km (Figure 4). The depth is a constant 10 m and the grid spacings, Δx and Δy are fixed at 1 km. Using the analytic solution, surface elevation and velocity are imposed at the southern boundary with $\Omega = 0.00001$, $T = 12$h, and $H_o = 50$cm. As can be seen in a comparison between the simulated and measured water level and velocity (Figure 5), the model is able to simulate this analytic test well.

10.3. Tidal forcing in a basin with linearly varying depth

To validate the wetting and drying scheme developed, a robust analytical test needs to be performed. Carrier and Greenspan [31] obtained the theoretical solution to wave propagation on a linearly sloping beach (Figure 6).

The one-dimensional non-linear shallow water equations can be written as

$$\frac{\partial \eta^*}{\partial t^*} + \frac{\partial}{\partial x^*}[(\eta^* + h^*)u^*] = 0 \tag{57}$$

$$\frac{\partial u^*}{\partial t^*} + u^* \frac{\partial u^*}{\partial x^*} + g \frac{\partial \eta^*}{\partial x^*} = 0 \tag{58}$$

where asterisks denote dimensional quantities, η is the water surface elevation above the mean water level, h is the still water depth which varies linearly with x, and u is the velocity in

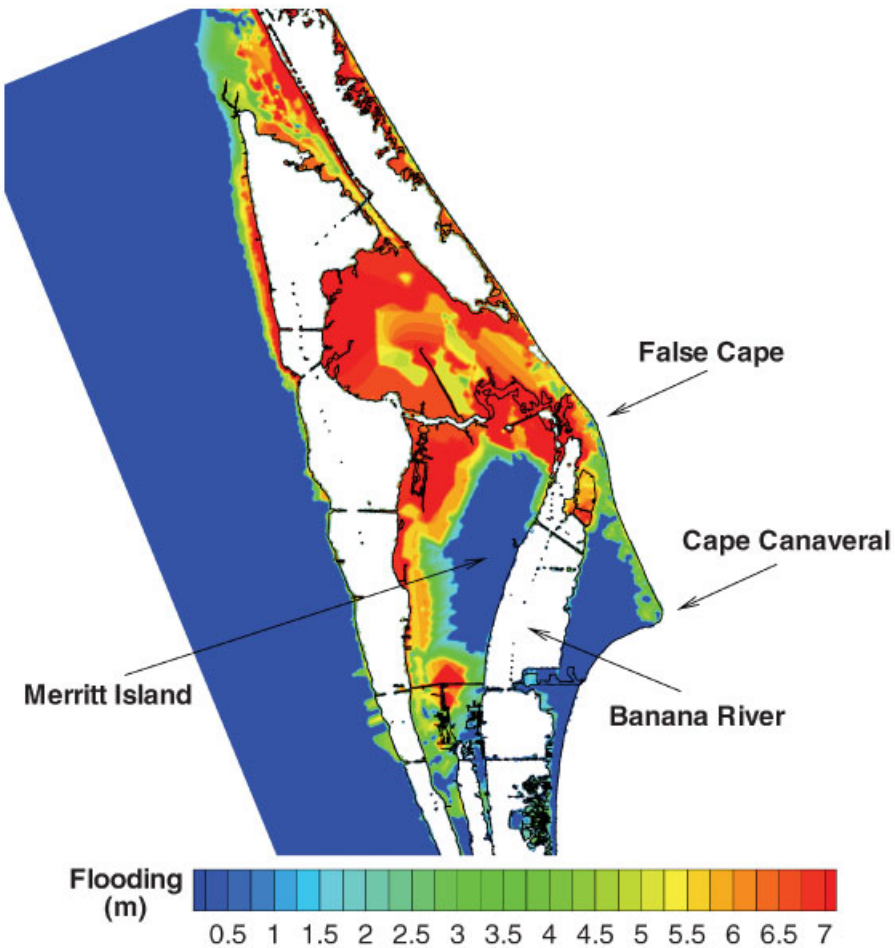


Plate 1. The simulated maximum depth of flooding during the passage of hypothetical Category 5 hurricane using the fine grid and making landfall at high tide. The surface area of flooded land and average depth of flooding are 584 km^2 and 5.04 m, respectively.

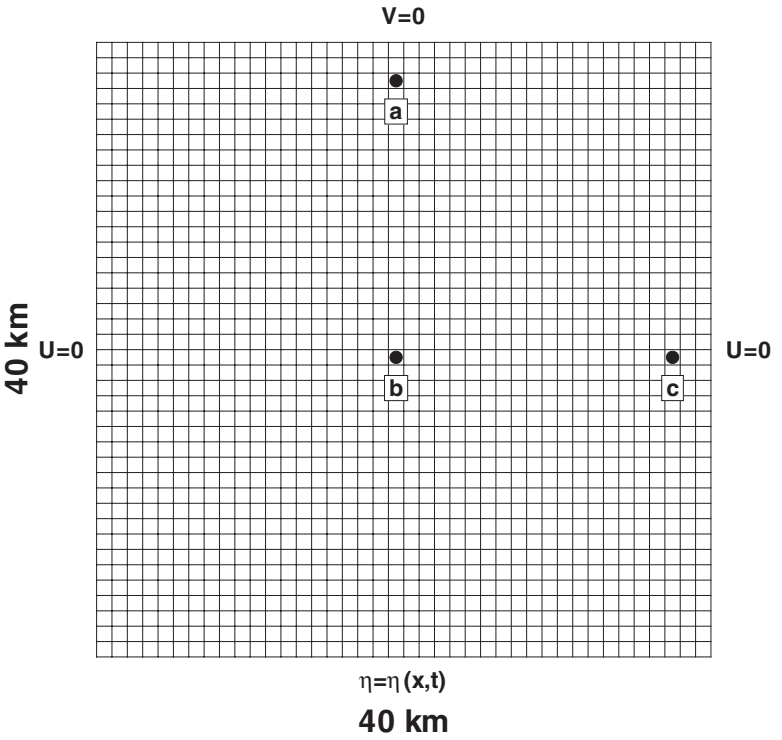


Figure 4. Grid system for Coriolis test. The three stations used for comparison are given the letters: a, b, and c.

the x-direction. Letting L be the characteristic length scale of the wave, the time and velocity scales can be defined as

$$T = \sqrt{\frac{L}{\phi g}} \tag{59}$$

$$u_o = \sqrt{\phi g L} \tag{60}$$

where ϕ is the beach angle. The equations are then non-dimensionalized using the following relations:

$$x = \frac{x^*}{L}$$

$$t = \frac{t^*}{T}$$

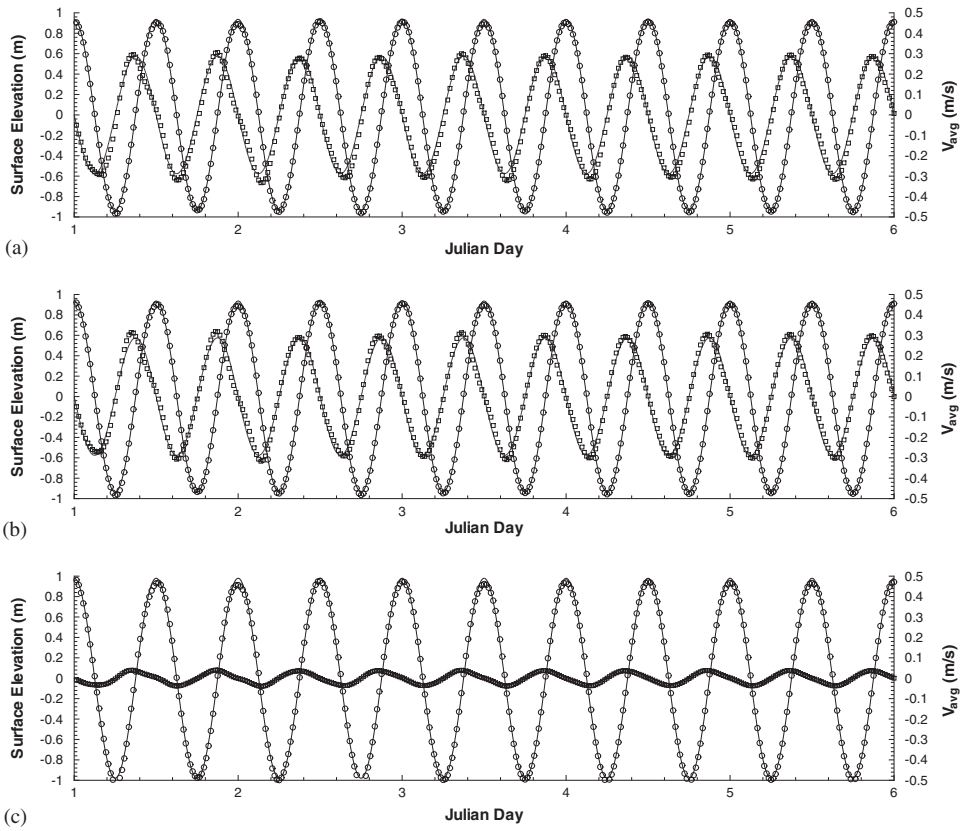


Figure 5. A comparison between analytic and simulated surface elevation and velocity at the three locations, a, b, and c. Solid lines indicate analytic solutions for surface elevation while dotted lines indicate analytic solutions for velocity in the y-direction. Circles indicate the simulated solution for surface elevation and squares indicate the simulated solution for velocity in the y-direction.

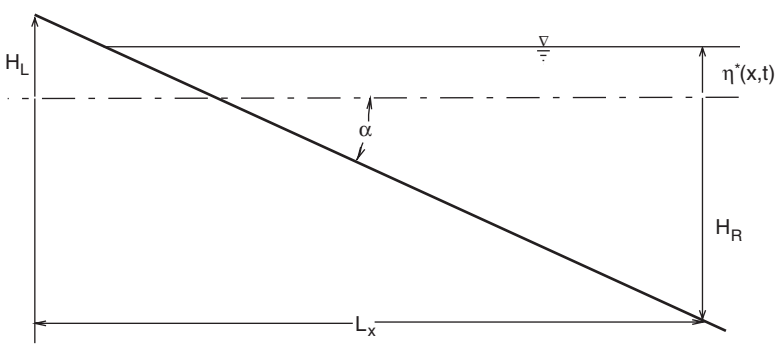


Figure 6. Wave propagating on a linearly sloping beach.

$$\eta = \frac{\eta^*}{\phi L} \tag{61}$$

$$h = \frac{h^*}{\phi L} = x$$

$$u = \frac{u^*}{u_o}$$

Defining $c = x + \eta$ and using the variables $\sigma = 4c$, $\lambda = 2(t - u)$, and a potential, $\varphi(\sigma, \lambda)$, [31] proposed the following expressions:

$$t = \frac{\lambda}{2} + u = \frac{\lambda}{2} + \frac{\varphi_\sigma}{\sigma} \tag{62}$$

$$x = \frac{u^2}{2} + c^2 + \frac{\varphi_\lambda}{4} = \frac{1}{2}\left(\frac{\varphi_\sigma}{\sigma}\right)^2 + \frac{\varphi_\lambda}{4} + \frac{\sigma^2}{16} \tag{63}$$

$$\eta = c^2 - x = \frac{\sigma^2}{16} - x = -\frac{\varphi_\lambda}{4} - \frac{\sigma^2}{16} \tag{64}$$

$$u = \frac{\varphi_\sigma}{\sigma} \tag{65}$$

$$c = \frac{\sigma}{4} \tag{66}$$

$$\varphi(\sigma, \lambda) = -8A_o J_0\left(\frac{\sigma}{2}\right) \sin\left(\frac{\lambda}{2}\right) \tag{67}$$

where A_o is an arbitrary amplitude parameter and J_0 is a zeroth order Bessel function of the first kind. The potential, φ represents a standing wave solution resulting from the perfect reflection of a unit frequency wave.

Equations (62) through (66) give t, x, η, u, and c parametrically in terms of σ and λ. In general, it is difficult to obtain direct functional relationships for η and u in terms of x and t. To evaluate $\eta(x,t)$ and $u(x,t)$ for a given x and t, Equations (62) through (66) must be solved numerically. For specific values of x and t, σ and λ are determined using a least squares method so that $\eta(x,t)$ and $u(x,t)$ are easily obtained from Equations (64) and (65), respectively.

The length of the basin, L_x, is 62 km and the width 10 km. The bottom slope, α, is 1:2500. The height above still water on the left side, H_L is 2 m and the depth below still water on the right side, H_R is 24 m. The period of the long wave is 1 h or a frequency, ω^*, of 0.001745 1/s. The characteristic length scale, L is defined by

$$L = \frac{\phi g}{(\omega^*)^2} = 1288 \text{ m} \tag{68}$$

which yields the velocity scale

$$u_o = \sqrt{\phi g L} = 2.25 \text{ m/s} \tag{69}$$

For the numerical simulation, a 620×5 rectangular grid ($\Delta x = 100\,\text{m}$, $\Delta y = 2000\,\text{m}$) is used along with a time step of $\Delta t = 30\,\text{s}$. The degree of implicitness, θ_1, is chosen to be 0.55 and the tolerance of the conjugate gradient solver is 10^{-6}. At $t = 0$ an initial wave form, as calculated by the theoretical solution, is applied to the surface elevation. For $t > 0$, the wave amplitude at the offshore boundary, $\eta^*(t)$ is given by

$$\eta^*(t) = \eta(t)\phi L \tag{70}$$

where $\eta(t)$ is the dimensionless value of the wave amplitude obtained from the theoretical solution. After three periods, the numerical solution of the standing wave is obtained and compared to the theoretical solution.

Figures 7 and 8 show comparisons between simulated and theoretical wave profiles over the length of the basin during 7 time instants of a half period. The model is able to simulate the wetting and drying of the shoreline well.

10.4. Conjugate gradient tolerance analysis

As previously discussed, an iterative solver is used to solve the 9-diagonal matrix (Equation (19)) for surface elevation which results from a semi-implicit or implicit discretization. This matrix is solved using Aztec [27], a parallel iterative solver. The iterative solving algorithm ceases if a residual (Equation (46)) is less than a specified tolerance. To determine the effect of changing this tolerance on the outcome of the numerical solver, several simulations were performed using different tolerances. The simulation chosen is the analytic test for tidal forcing with Coriolis discussed previously. Comparisons between the simulated water level and velocities using the very small default tolerance (10^{-6}) and larger ones are shown in Table III. The results show that a tolerance as large 10^{-4} changes the results by less than $10^{-4}\%$.

10.5. Parallel timing analysis

To determine how well the model performs in parallel, simulations are performed using various processor and simulation configurations on the explicit, semi-implicit and implicit model formulations. In addition, the conjugate gradient algorithm in Aztec [27] can be modified; thus, affecting the simulation time. The simulation chosen as a basis for experimentation is the simple wind setup test described previously. Figure 9 shows the grid configuration for a multi-processor case with the x-direction split among four processors.

10.5.1. Explicit mode. In general, explicit discretization of equations yields equations which are easily solved in parallel. Difficulty and loss of speed occurs only at the end of a given time step when information must be passed to other processors. However, due to stringent time step limitations ($\Delta t \leqslant \Delta x/u$), explicit formulations are difficult to use in practical simulations which require fast simulation times and small grid spacings. Parallel speedup for various grid sizes using a 1 s time step is shown in Figure 10. These results show how the speedup improves with a larger number of grid cells in the x-direction; however, as the number of cells in the y-direction gets larger (the direction through which data is passed to other processors), the speedup decreases.

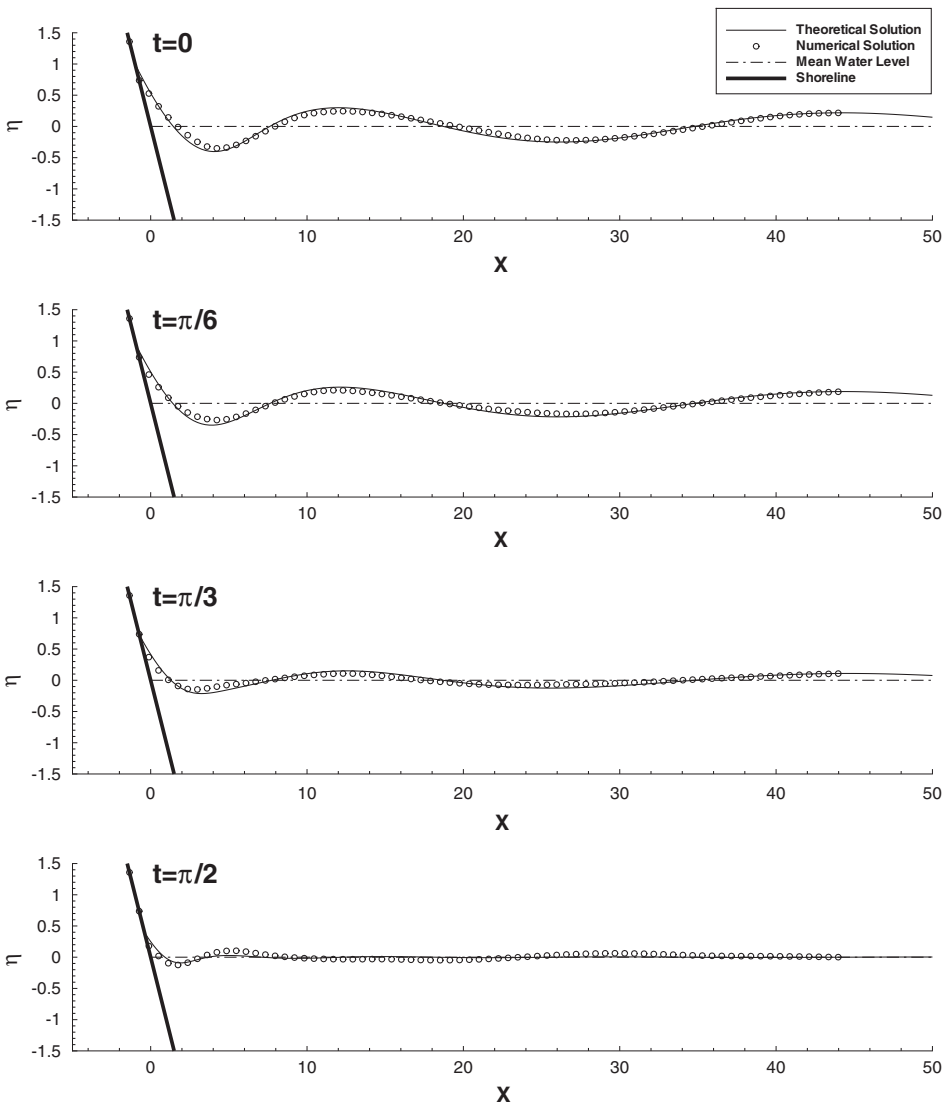


Figure 7. Non-dimensional wave profiles as predicted by theory and the numerical model for times $t = 0$ through $t = \pi/2$.

10.5.2. Semi-implicit and implicit modes. While implicit discretization is more difficult to parallelize, it allows much larger time steps to be used during a simulation. Analysis of the parallel speedup for various grid configuration using the implicit modes (Figure 11(a) and 11(b)) show that the model's speedup is not a function of either N_x or N_y explicitly, but as a function of the total number of grid cells ($N_x \times N_y$). In other words, the speedup characteristics of the 1600×25 configuration is nearly the same as the speedup characteristics of the 200×200 configuration.

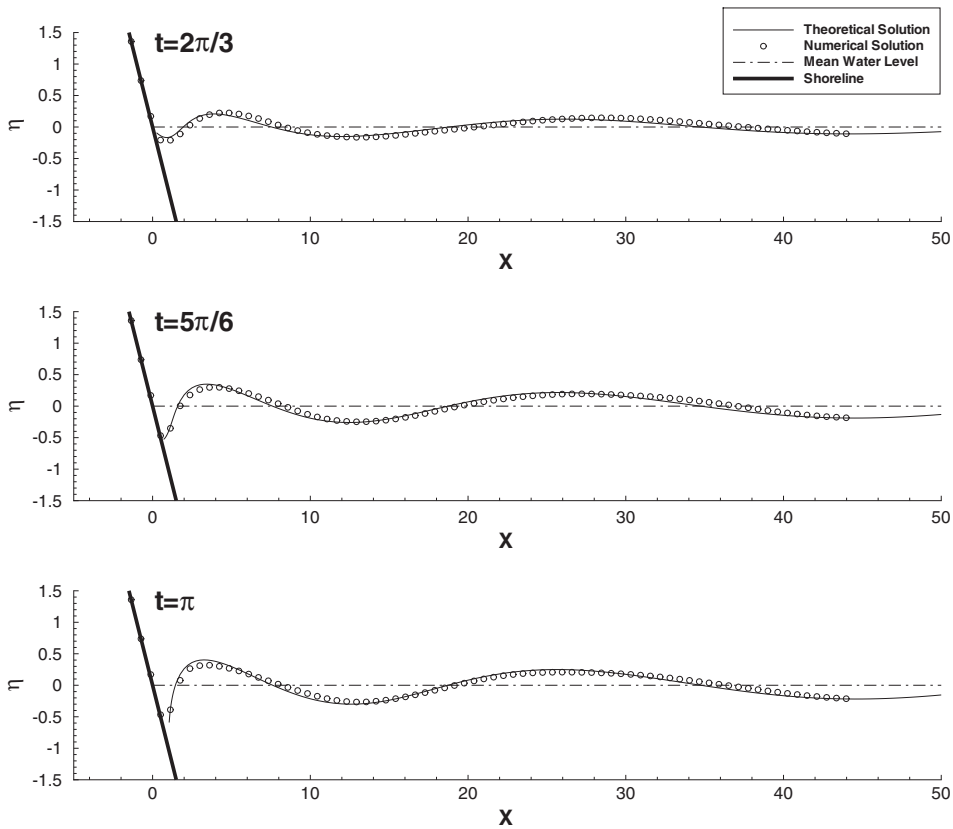


Figure 8. Non-dimensional wave profiles as predicted by theory and the numerical model for times $t = 2\pi/3$ through $t = \pi$.

In addition to varying simulation parameters, parallel timing simulations were performed using the various configurations available with Aztec's iterative solver (Figure 12). The conjugate gradient squared algorithm showed significantly better speedup characteristics than either the TFQMR or BICGSTAB algorithms; however, none of the scaling algorithms showed significantly better speedup characteristics than any of the other scaling algorithms. As the tolerance of the iterative solver is enlarged, fewer iterations are required thus reducing required inter-processor communications. This reduction in communication thus improves the speedup characteristics of the model.

As can be seen from the previous results, efficiencies increase with larger grid sizes. Figure 13 shows a series of simulations performed on large grid systems illustrating this point. For the 200×6400 (1.28 million computational cells) and 20 processors, a speedup of 15.10 is reached.

10.5.3. Note on linear decomposition. Currently, the PEM only allows the grid system to split entirely in the x-direction or entirely in the y-direction. In general, a two-dimensional domain

Table III. The RMS difference ($/10^{-4}$) between the surface elevation, η, u-averaged, U, and v-averaged, V, velocities calculated with a conjugate gradient tolerance of 10^{-6} and the results calculated with tolerances of 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5}. Results shown are for the 5 day simulation used to compare with the analytic test for Coriolis and are given in units of m and m/s, respectively.

		Tolerance				
Station	Variable	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
a	η	5.41	1.35	0.57	0.17	0.00
b	U	1.49	0.29	0.12	0.00	0.00
c	V	3.07	0.71	0.24	0.17	0.00
a	η	4.52	1.25	0.47	0.17	0.00
b	U	1.20	0.37	0.29	0.24	0.00
c	V	5.00	1.34	0.49	0.37	0.00
a	η	4.57	1.28	0.49	0.12	0.00
b	U	2.55	0.59	0.12	0.24	0.00
c	V	5.08	1.45	0.60	0.51	0.00

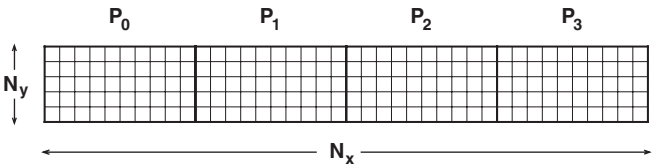


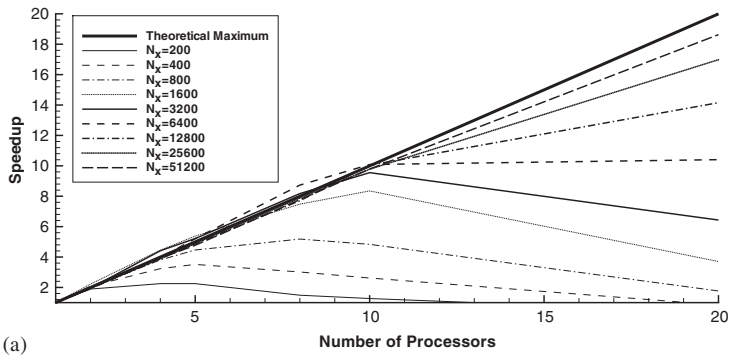
Figure 9. Four processor grid configuration for the simulations used to calculate speedup with the PEM. The number of cells in the x- and y-directions are N_x and N_y, respectively.

decomposition configuration would be more efficient. For example, assuming an overall grid size of 1400×640 and 2 ghost cells, a 20×1 configuration needs to communicate 1.8 times more information than a 4×5 configuration. Future versions of the PEM will include the ability to handle two-dimensional domain decompositions and will improve the speedup characteristics of the PEM for both the explicit and implicit modes.

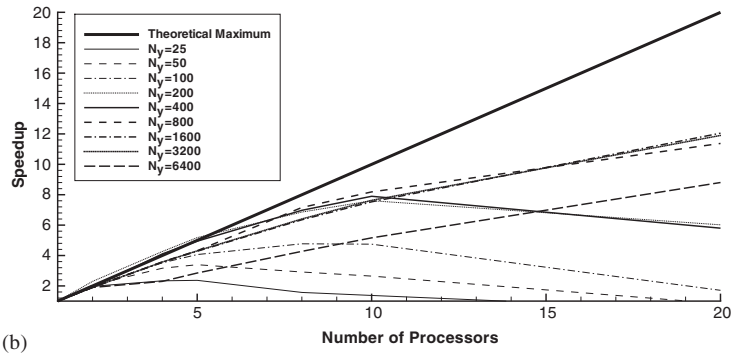
11. APPLICATION

In the following section, an typical PEM application is presented: the storm surge caused by a hypothetical hurricane making landfall in the Indian River Lagoon, Florida. A detailed model calibration and verification study of the Indian River storm surge simulations will be reported separately.

The Indian River Lagoon (Figure 14) is an estuary located in Brevard County on the Atlantic coastal of central Florida. The lagoon is approximately 240 km long and extends from Ponce de Leon Inlet in the north to Jupiter Inlet in the south. The lagoon is 2–4 km



(a)



(b)

Figure 10. Parallel speedup using the explicit PEM and various grid configurations. The time steps are 1 s and the number of cells in the opposite direction is fixed at (a) $N_y = 25$ and (b) $N_x = 200$.

wide with an average depth of 2 m outside of the Intracoastal Waterway (ICW). The lagoon receives fresh water from numerous natural creeks, rivers and man-made canals.

For the IRL storm surge simulations, a 1400×640 grid was developed for the PEM. The entire grid system was then split into a 20 processor configuration to correspond with a 20 processor Beowulf Cluster [32]. Each node of the cluster consists of a dual processor 450 MHz Dell PowerEdge 1300 with 256 MB of main memory. The IRL PEM grid system has a constant Δx and Δy of 125 m and was divided evenly in the x-direction such that each of the 20 processors solved a 70×640 section of the overall grid (Figure 15). The grid system was designed to both include as much of the land area to the west of the lagoon as practically possible for simulation of flooding due to storms and include as much of the offshore area as possible to minimize the effect of the boundary on model results. Grid elevation was specified using high resolution bathymetric surveys and five foot Digital Elevation Model (DEM) topographic contours provided by the St. Johns River Water Management District (SJRWMD). Although the Cartesian grid system can only fit the complicated shoreline geometry in a 'stair-step' fashion, the very fine grid resolution is sufficient to resolve all but the smallest coastal features. A curvilinear grid version of the PEM capable of boundary-fitting the shoreline is being developed.

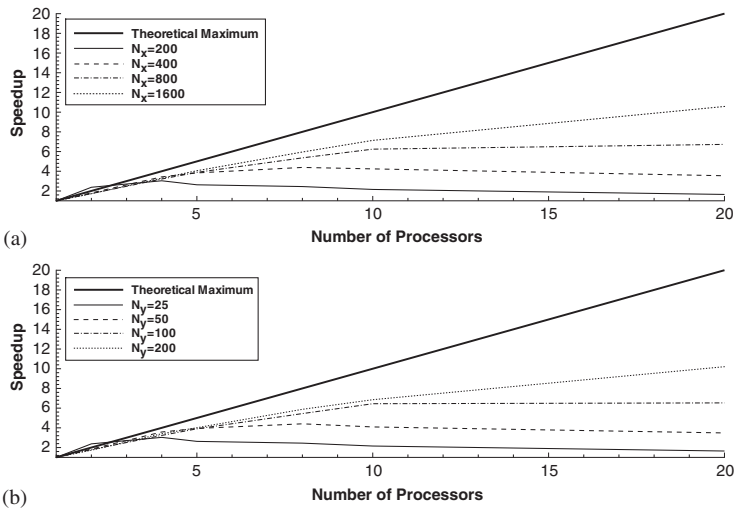


Figure 11. Parallel speedup using the semi-implicit PEM for various grid sizes where (a) $N_y = 25$ and (b) $N_x = 200$.

Using the historically most probable approach direction of a landfalling hurricane in Brevard County [33], 130°, a hypothetical hurricane track was developed (Figure 16). The hypothetical storm was designed to have a translational velocity of 10 knots, a radius to maximum wind speed of 20 n mi and a ratio of surface wind velocity to gradient wind velocity of 1.4. These properties yield a maximum wind speed 160 mph making it a large Category 5 [34] (wind speed greater than 155 mph) hurricane. The impact point was specified to be at False Cape (80 34 34 W, 28 35 12 N).

To simulate the storm surge in the lagoon due to the landfalling hypothetical hurricane, the PEM model was setup to simulate 6 days in 1999, 3 October through 9 October, using a time step of 15 min. The hurricane made landfall approximately halfway through the simulation period, 6 October (10:57 pm). Using water level measured at Sebastian Inlet, measured tidal forcing was specified at the ocean open boundary for the entire simulation with the hurricane making landfall exactly at high tide. The wind and air pressure fields were generated using the simple storm model contained within the PEM.

Plate 1 illustrates the amount of flooding caused by the hurricane making landfall at high tide. This simulation shows the large amount of flooding which would be associated with landfall of a major hurricane. All of the northern part of Merritt Island is flooded along with a significant portion of the river front in the northern IRL. More flooding is seen on the IRL itself rather than areas closer to the ocean such as Banana River.

The six day simulation was performed on the 20-processor Beowulf Cluster described previously and was completed in 13.0 h (11 $\times$ real time). The parallel solution of the 9-diagonal coupled system of equations took 82.2% of the total computational time, while the parallel ELM routine, used to calculate the non-linear and diffusion terms, took approximately 17.5% of the total simulation time.

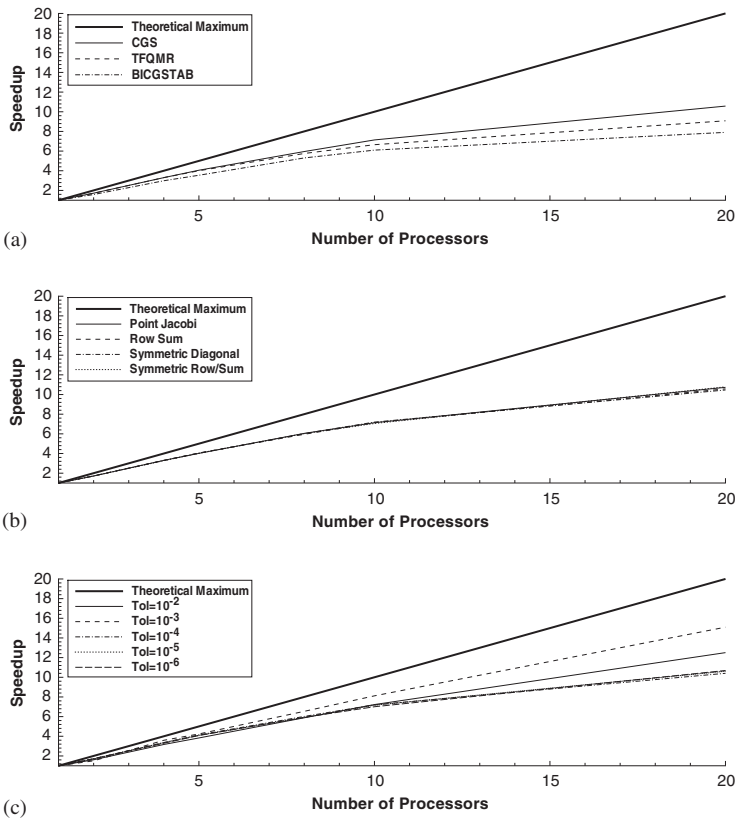


Figure 12. Parallel speedup using the semi-implicit PEM and various iterative solve configurations. The algorithm, scaling and tolerance are (a) Row Sum and 10^{-6}, (b) CGS and 10^{-6} and (c) CGS and Row Sum.

12. CONCLUSIONS

A new parallel storm surge model, the Parallel Environmental Model (PEM), is developed and verified through comparisons with analytic solutions. The PEM is a 2-D vertically averaged, wetting and drying numerical model. The propagation, bottom friction and Coriolis terms are solved semi-implicitly resulting in a 9-diagonal system of equations which is solved using the parallel solver, Aztec. The advection and diffusion terms are solved with a new parallel Eulerian–Lagrangian scheme developed for this study. It is noted that the parallel ELM algorithm developed could be easily adapted by developers of other parallel ocean or estuary models. Storm boundary conditions are based on a simple exponential decay of pressure from the center of a storm.

The PEM was developed specifically for use on parallel computer systems and will function accordingly in either explicit of implicit modes. The model was verified with various analytic solutions. As an example application, the storm surge caused by a landfalling Category 5 hurricane was simulated. When landfall occurs directly at high tide, nearly 5 m of flooding

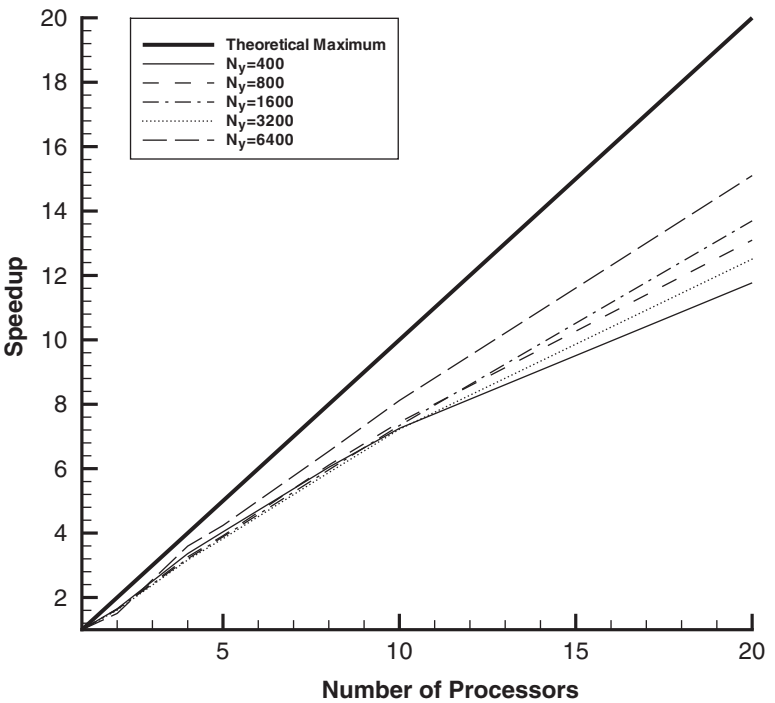


Figure 13. Parallel speedup using the semi-implicit PEM for large values of N_y where $N_x = 200$ and $\Delta t = 900$. Results are calculated for one time step.

Figure 14. Location of the Indian River Lagoon study area.

was simulated. The simulation presented was designed to show a typical application of the PEM, detailed model calibration and verification of the Indian River storm surge simulations will be reported separately.

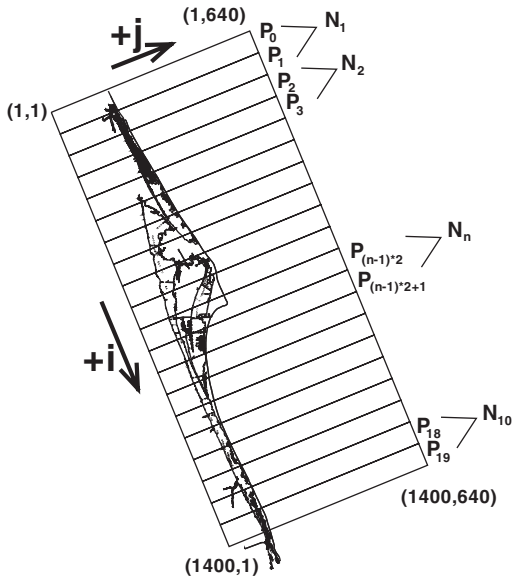


Figure 15. The IRL grid system developed for the PEM ($\Delta x = \Delta y = 125$ m).

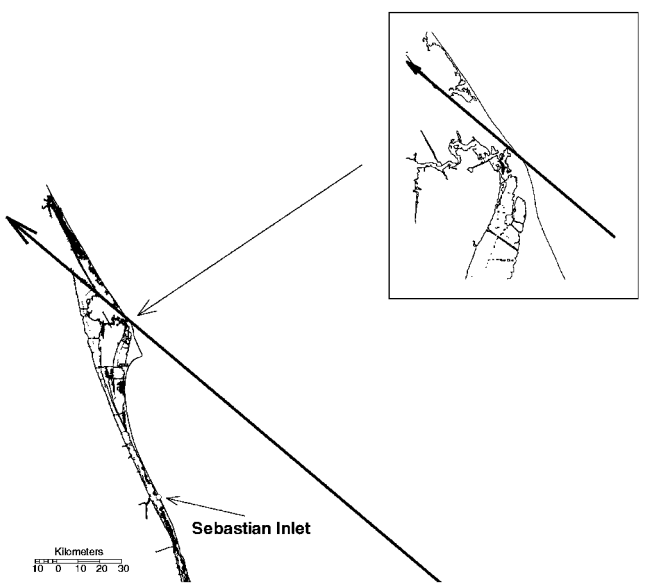


Figure 16. The track of the hypothetical hurricane making landfall at False Cape, just north of Cape Canaveral.

The overall model performed well with good parallel speedup and efficiency characteristics; however, for the example simulation, nearly 20% of the model's execution time was spent in the parallel ELM algorithm. As the algorithm tracks streak lines backward, they may enter the domain of other processors. When this occurs, a message must be sent the corresponding processor so that it may continue the streak line backwards. This message passing between processors is the slowest part of the algorithm. It is hypothesized that this phase can be sped up by using the processors 'ghost cells' in streak line calculation. Currently, the parallel ELM algorithm sends a message to its neighbour even if only one cell outside the boundary is needed for streak line calculation.

While the current two-dimensionality of the PEM does limit its applications, the model is highly amenable to modification and conversion of the model to three-dimensions as well as the addition of salinity and temperature transport to the model is progressing. Future versions of the PEM will also include the addition of a wave model, better storm formulation as well as coupling with GIS to produce quantitative analysis of a storm's impact such as numbers of home and road's flooded.

ACKNOWLEDGEMENTS

The high performance modelling work is supported by the National Center for Environmental Research and Quality Control, USEPA and the University of Florida.

REFERENCES

1. Sheng YP. Development of a preliminary hydrodynamics and water quality model of the Indian River Lagoon. Final Report to the St. Johns River Water Management District, Coastal and Oceanographic Engineering Department, University of Florida, Gainesville, FL, 1997.
2. Sheng YP. A framework for integrated modeling of coupled hydrodynamic-sedimentary-ecological processes. In *Estuarine and Coastal Modeling*, vol. 6. American Society of Civil Engineers, 2000; 350–362.
3. Beck R, Dean S, O'Keefe M, Sawdey A. A comparison of data-parallel and message-passing versions of the Miami Isopycnic Coordinate Ocean Model MICOM. *Parallel Computing* 1995; **21**(10):1695–1720.
4. Luong P, Breshears CP, Gabb HA. Execution and load-balance improvements in the CH3D hydrodynamic simulation code. *Technical Report* 00-07, U.S. Army Engineer Research and Development Center, Major Shared Resource Center, Information Technology Laboratory, Vicksburg, Mississippi, February 2000. http://www.wes.hpc.mil/pet/tech_reports/reports/report_index_bd.htm.
5. Zhu J, Johnson B, Bangalore P, Huddleston D, Skjellum T. On the parallelization of CH3D. *Technical Report 98-07*, U.S. Army Corps of Engineers Waterways Experiment Station, Major Shared Resource Center, Information Technology Laboratory, Vicksburg, Mississippi, 1998. http://www.wes.hpc.mil/pet/tech_reports/reports/report_index_bd.htm.
6. Zhu J, Johnson B, Bangalore P, Huddleston D, Skjellum T. On the parallelization of CH3D. In *Proceedings of the International Water Resources Engineering Conference*, vol. 2, Abt SR, Young-Pezeshk J, Watson CC (eds). American Society of Civil Engineers, 1998; 1108–1113.
7. Boukas LA, Mimikou NT, Missirlis NM, Mellor GL, Lascaratos A, Korres G. The parallelization of the Princeton Ocean Model. In *EURO-PAR'99 Parallel Processing*, Amestoy P, Berger P, Dayde M, Duff I, Fraysse V, Giraud L, Ruiz D (eds), Lecture Notes in Computer Science, vol. 1685, Springer: Berlin, Germany, 2000; 1395–1402.
8. Davis JR, Sheng YP. Parallelization of the CH3D integrated modeling system. *in review*, 2003.
9. Sheng YP, Davis JR, Sun D, Qiu C, Christian D, Park K, Kim T, Zhang Y. Application of an integrated modeling system for estuarine and coastal ecosystems to Indian River Lagoon, Florida. In *Estuarine and Coastal Modeling*, vol. 7. American Society of Civil Engineers, 2002; 329–343.
10. Pielke RA. *The Hurricane*. Routledge: New York, 1990.
11. Sheng YP. Mathematical modeling of three-dimensional coastal currents and sediment dispersion: Model development and application. *Technical Report CERC-83-2*, Aeronautical Research Associates of Princeton, Princeton, New Jersey, 1983.
12. Van Dorn W. Wind stress on an artificial pond. *Journal of Marine Research* 1953; **12**(3):249–276.

13. Hsu SA. A proper wind-stress drag coefficient formulation for computer modeling of seas and coastal regions. In *Computer Modeling of Seas and Coastal Regions II*, Brebbia CA, Traversoni L, Wrobel LC (eds). Computational Mechanics Publications, 1995; 413–420.
14. Smith SD, Banke EG. Variation of sea-surface drag coefficient with wind speed. *Quarterly Journal of the Royal Meteorological Society* 1975; **101**(429):665–673.
15. Garratt JR. Review of drag coefficients over oceans and continents. *Monthly Weather Review* 1977; **105**:915–929.
16. Staniforth A, Cote J. Semi-Lagrangian integration schemes for atmospheric models—A review. *Monthly Weather Review* 1991; **119**(9):2206–2223.
17. Casulli V, Cheng RT. Semi-implicit finite difference methods for three-dimensional shallow water flow. *International Journal for Numerical Methods in Fluids* 1992; **15**:629–648.
18. Oliveira A, Fortunato AB, Baptista AM. Mass balance in Eulerian–Lagrangian transport simulations in estuaries. *Journal of Hydraulic Engineering* 2000; **126**(8):605–614.
19. Behrens J. An adaptive semi-Lagrangian advection scheme and its parallelization. *Monthly Weather Review* 1996; **124**:2386–2395.
20. Malevsky AV, Thomas SJ. Parallel algorithms for semi-lagrangian advection. *International Journal for Numerical Methods in Fluids* 1997; **25**(4):455–473.
21. Thompson EF, Cardone VJ. Practical modeling of hurricane surface wind fields. *Journal of Waterway*, *Port*, *Coastal and Ocean Engineering* 1996;**122**(4):195–205.
22. Vickery PJ, Twisdale LA. Prediction of hurricane wind speeds in the United States. *Journal of Structural Engineering* 1995; **121**(11):1691–1699.
23. Vickery PJ, Skerlj PF, Steckley AC, Twisdale LA. Hurricane wind field model for use in hurricane simulations. *Journal of Structural Engineering* 2000; **126**(10):1203–1221.
24. Holland GJ. An analytic model of the wind and pressure profiles in hurricanes. *Monthly Weather Review* 1980; **108**(8):1212–1218.
25. Wilson BW. Hurricane wave statistics for the Gulf of Mexico. *Technical Memorandum* 98, Department of the Army Corps of Engineers, June 1957.
26. Gropp W, Lusk E. *User's Guide for mpich*, *A Portable Implementation of MPI*: *Version 1.2.2*. Mathematical and Computer Science Division, Argonne National Laboratory, University of Chicago, 2001. ftp://ftp.mcs.anl.gov/pub/mpi/userguide.ps.
27. Tuminaro RS, Heroux M, Hutchinson SA, Shadid JN. Official Aztec User's Guide—Version 2.1. *Technical Report* SAND99-8801J, Sandia National Laboratories, Albuquerque, New Mexico 87185, November 1999. http://www.cs.sandia.gov/CRF/aztec1.html.
28. Shadid JN, Tuminaro RS. Sparse iterative algorithm software for large-scale MIMD machines: An initial discussion and implementation. *Concurrency*, *Practice and Experience* 1992; **4**(6):481–489.
29. Carney S, Heroux M, Li G. A proposal for a sparse BLAS toolkit. *Technical report*, Cray Research Inc., Eagen, Minnesota, 1993.
30. M. Rahman. Analytical solutions for tidal propagation in a rectangular basin. *Advances in Water Resources* 1983; **6**:44–53.
31. Carrier JR, Greenspan HP. Water waves of finite amplitude on a sloping beach. *Journal of Fluid Mechanics* 1958; **4**:97–109.
32. Davis JR, Sheng YP. High performance estuarine and coastal environmental modeling: Part II. In *Estuarine and Coastal Modeling*, vol. 7. American Society of Civil Engineers, 2002; 479–490.
33. Dean RG, Chiu TY. *Combined total storm tide frequency analysis for Brevard County*, *Florida*. Beaches and Shores Resource Center, Institute of Science and Public Affairs, Florida State University, Tallahassee, Florida, May 1986.
34. Simpson RH, Riehl H. *The Hurricane and its Impact*. Louisiana State University Press: Baton Rouge, Louisiana, 1981.

