An efficient parallel model for coastal transport process simulation

Onyx Wing-Hong Wai a,*, Qimiao Lu b

a Department of Civil and Structural Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
b Baird and Associates, 627 Lyons Lane, Suite 200, Oakville, Ontario, Canada L6J 5Z7

Received 14 April 1999; received in revised form 23 December 1999; accepted 12 January 2000

Abstract

A three-dimensional (3D) parallel model for efficient simulation of sediment–water transport processes in coastal regions is introduced in this paper with a main focus on the parallel architecture of the model. The model’s parallel efficiency is maximized in two steps. First, a fully parallelizable hybrid operator splitting numerical technique is applied to discretize the governing equations of the model. Within one computation time step, the technique makes use of an explicit Eulerian–Lagrangian scheme to deal with the advection, and a finite element method and a finite difference method to handle the horizontal and vertical diffusion, respectively. Second, the model parallelization is established according to a mixed strip/patch domain decomposition method and master–slave communication approach on multiple instruction multiple data (MIMD) machines. In order to test the model efficiency and accuracy, two real-life applications were conducted on a MIMD parallel computer, IBM SP2/900. The computed results agree reasonably well with field measurements. A new iterative matrix solving technique developed especially for coastal process calculations, namely the Gauss–Jacobi (GJ) iteration method, has also been implemented in the two applications. With the new technique, the performance of the parallel model can be improved five-fold for each extra processor used. The performance leap greatly increases the engineering relevance of the model, for instance, predicting the long-term fate of dredged material disposed in a dynamic environment. © 2000 Elsevier Science Ltd. All rights reserved.

1. Introduction

The time scales of coastal processes, such as water movement and transport processes of sediments, vary widely from a few seconds to days or even years. Hence, simulation periods for transport processes in estuaries and coastal regions should at least cover a tidal event (typically about 12–24 h), or a neap-spring tidal event (fortnightly), or even a seasonal event. On the other hand, the time step interval should be small enough in order not to overshoot the short life activities such as gravity waves and bottom sediment resuspension. This implies that a large number of time steps and huge amount of computation time are required in order to achieve a meaningful simulation. Coastal transport models built for various degrees of physical complexity can be found in the literature. Key issues concerned in these models usually are how the models account for the (1) spatial dimension, (2) flow field coupling, (3) driving forces, (4) treatment of the horizontal dispersion and vertical diffusivity, (5) bedload transport, (6) bed erosion and deformation, (7) cohesiveness of sediment, (8) stratification due to sediment, temperature and salinity, and so forth. A list of 2D numerical models was summarized by van Rijn [29]. Several finite difference 3D sediment transport models that have taken into account of the combined effect of waves and currents on the bottom shear stress were developed by Sheng [25] and others. A large-scale study of sediment transport patterns in Hong Kong waters was presented by Wai et al. [31]. There are also a few commercial 3D sediment transport models such as Delft Hydraulics’ Delft3D, HydroQual’s SEDZL, Danish Hydraulic Institute’s MIKE-3, ERM’s TELEMAC-3D, the Rand Corp.’s SED-8 and so forth. Large-scale simulations such as estuarine 3D sediment-water coupled simulations remain relatively rare and this is mainly due to lack of computing power and insufficient 3D information. To increase the computing performance so that more transport parameters can be accommodated/considered and large 3D simulations can be conducted for gaining physical insights and engineering relevance, parallel computation is uniquely positioned to play a major role [5].
In parallel computations, many processors are working simultaneously to solve one problem. A large-scale problem can be partitioned into several smaller problems that can be solved by separate processors. Communication is required to exchange messages between processors. The efficiency of parallel computations depends on (1) the flexibility of numerical method for parallelization, (2) the solver of linear algebraic systems and (3) communication strategy and data structure. A few papers presenting parallel algorithms in 3D hydrodynamic and transport models are found in the recent literature (e.g. [3,4,7,14,20,27,28]). Song et al. [28] compared the flexibility of the explicit finite difference method (FDM) and the alternating direction implicit (ADI) method for parallelization. The explicit methods are straightforward to parallelize and can achieve a high degree of parallelism. However, the stability is generally poor for the explicit methods which limits the time step and reduces their applicability. The ADI method is difficult to parallelize because the solutions in the x and y directions of the tridiagonal system are alternately dependent on each other. For these reasons, Goede [6] developed an unconditionally stable two-stage alternating operator implicit (AOI) method for a 3D curvilinear grid model and a multiple block implementation is used for the model parallelization. In AOI, the parallel computations are divided into two stages. A large number of tridiagonal systems are solved in the first stage. With the multiple block implementation, the solution of these systems can be vectorized. In the second stage the velocity components are eliminated so only one equation remains for solving the tidal elevations. The equation is solved with a preconditioned conjugated gradient method combined with a smoothing technique, which is effective for parallel computers. The AOI method is more efficient and suitable for parallel computations than the conventional ADI method. Kok [13] employed an implicit Odd–Even Line Hopscotch (OELH) method in his 3D transport FDM model. In the OELH method, the solution system is split into two stages. In the first stage, only the odd grids are solved by replacing all components corresponding to the even grids with zero. In the second stage, the calculations are only limited to the even grids by setting all components of the odd grids to zero. This method can be straightforwardly vectorized for parallel computations and is proven to be highly efficient for integrating the space-discretized transport equations on the CRAY-type computers. All the parallel models mentioned above are discretized with FDM, which requires a regular grid structure that is difficult to fit to complex irregular boundaries.

Implicit finite element systems are a better choice for irregular boundary geometry like coastlines. However, the irregularity will give rise to a set of irregular algebraic equations and solving the irregular matrix system is very timely. Hence, the important issue in finite-element model parallelization is how to efficiently solve the algebraic equations with parallel computations. There are a few methods available for solving...
sparse algebraic matrices, for example, the Gauss–Seidel (GS) method, Jacobi iteration method and conjugate gradient (CG) iteration method. Though the conventional GS eliminating method is stable and accurate for sequential computing, it is impossible to parallelize using the normal lexicographic ordering because the elimination method is data dependent. Iteration methods, such as the Jacobi method and CG method, have no data dependency and are the ideal candidates for parallel matrix solvers. Smith and Fides [26] compared the currently available iteration methods, such as the Jacobi method, the Red/Black Gauss–Seidel (RBGS) method [23,34], and the preconditioned CG method, and showed that it is not obvious which method is faster on a parallel computer. The CG and RBGS iterations are slightly faster than other methods in solving large sets of algebraic equations with sparse coefficient matrices. However, these methods become inefficient in solving algebraic systems with non-symmetric matrices that are often encountered in coastal process modeling with substantial land boundary variation due to flooding and drying within a tidal cycle.

The focus of this paper is centered on the parallel architecture implementation and the model applications. The numerical formulation of the parallel model is only discussed in brief in the next section.

2. Numerical specification

2.1. Governing equations

The standard governing equations for coastal transport processes with the shallow water assumption are the continuity equation, the momentum equations, sediment transport equation and the bed deformation equation. The continuity equation and the Navier–Stokes equations for shallow waters with the assumption of hydrostatic pressure are shown as follows:

\[
\frac{\partial u_j}{\partial x_j} = 0, \quad (1)
\]

\[
\frac{\partial u_i}{\partial t} + v_j \frac{\partial u_i}{\partial x_j} - f \delta_{ij} v_j + g \frac{\partial \zeta}{\partial x_i} \left( \epsilon_{ij} \frac{\partial u_j}{\partial x_j} \right), \quad (2)
\]

where \( i = 1, 2, \quad j = 1, 2, 3, \quad t \) is time and \( x_i = (x, y, z) \) are spatial coordinates in the Cartesian coordinate system; \( v_j \) is the component of flow velocity in the \( x_j \) direction; \( \zeta \) is the water level from the mean sea level; \( f \) is the Coriolis parameter, \( f = 2\Omega \sin \theta \), \( \Omega \) the angular frequency of the earth rotation (s\(^{-1}\)), \( \theta \) the latitude of the computed region (\(^\circ\)); \( \epsilon_{ij} \) is the eddy viscosity coefficient for clear water in the \( x_i \) direction; \( \delta_{ij} \) is the Kronecker delta. Note that with the shallow water assumption Eq. (2) reduces to hydrostatic pressure equation in the vertical direction.

From the law of mass balance and eddy-diffusivity concept, the governing equation for sediment–water mixture of low concentration in the Cartesian coordinate system is written as follows:

\[
\frac{\partial c}{\partial t} + v_j \frac{\partial c}{\partial x_j} - \frac{\partial (\alpha_s c)}{\partial x_3} = \frac{\partial}{\partial x_j} \left( \epsilon_{xj} \frac{\partial c}{\partial x_j} \right), \quad (3)
\]

where \( c \) is sediment concentration (kg/m\(^3\)), \( \epsilon_{xj} \) a sediment diffusivity coefficient in the \( x_i \) direction (m\(^2\)/s), and \( \alpha_s \) the settling velocity of the sediment particles (m/s).

Sediment exchange in the water column denoted as \( q_s \) is defined as

\[
q_s = \alpha_s c + \epsilon_{x3} \frac{\partial c}{\partial x_3}. \quad (4)
\]

This expression represents the net sediment flux in the vertical direction which is the difference between the downward sediment flux, \( \alpha_s c \), and the upward sediment flux, \( \epsilon_{x3} \frac{\partial c}{\partial x_3} \). The parameters, such as the turbulence damping factor, near-bed equilibrium reference concentration and so forth, used for the deposition and entrainment associated with the complex sediment dynamics are mentioned in detail by Wai et al. [30].

2.2. Equation splitting

There are basically no compilers that enable good parallel performance from an inherently nonconcurrent computational procedure. The first step toward high performance parallel computation is to develop or adopt an efficient and parallelizable numerical method for the governing equations so that stable and accurate results can be obtained. Fisher and Patera [5] have identified two basic numerical strategies for parallel computations of the incompressible Navier–Stokes problem: the full-implicit temporal discretization approach; and the semi-implicit fractional time-stepping approach. Since the full-implicit approach requires considerable operation and memory for 3D calculations, the fractional step approach is adopted for the present 3D model. Lu and Wai [17] have developed a hybrid operator splitting (HOS) method for 3D shallow water flows, in which the method splits a time step to three fractional steps according to different physical significance such as advection, horizontal diffusion and vertical convection/diffusion. For brevity only the transport equation split and discretized with the fractional step approach will be demonstrated here. Detailed splitting formulation for the continuity and momentum equations is referred to Lu and Wai [17].

Before the implementation of the splitting procedure, the transport equation in Cartesian coordinates \((x_1, x_2, x_3)\) is transformed to the topography following \(\sigma\)-coordinate system \((x_1, x_2, \sigma = (x_3 + h)/H)\) in which \(h\) is the mean water depth and \(H = \zeta + h\). The transformed equation is then split into three subsystems that
represent the advection, the horizontal diffusion and vertical diffusion accompanying with sediment settling.

In the first fractional step, the concentration is solved in the present of the advection term only as shown in Eq. (5):

$$\frac{c^{n+1/3} - c^n}{\Delta t} + u_j^n \frac{\partial c^n}{\partial x_j} = 0. \tag{5}$$

Here the superscript denotes the variable in fractional time step and $\Delta t$ is the time step. The horizontal diffusion term representing the diffusive transport of the sediment induced by the horizontal difference of concentration is included in the second fractional step as shown in Eq. (6).

$$\frac{c^{n+2/3} - c^{n+1/3}}{\Delta t} = \frac{\partial}{\partial x_i} \left( \frac{\partial c^{n+1/3}}{\partial x_i} + \frac{\partial c^n}{\partial x_i} \right) + \text{HOT.} \tag{6}$$

The last term, HOT, in Eq. (6) is the higher-order terms resulted from the $\sigma$-coordinate transformation.

The variation of sediment concentration in the water column is primarily due to the balance between the vertical eddy diffusion and sediment settling. When the amount of suspended sediment being settled by gravity is equal to that being kept in suspension by vertical diffusion, an equilibrium state is reached. Hence, the two terms governing these processes are solved together in the last fractional step, written as

$$\frac{c^{n+1} - c^{n+2/3}}{\Delta t} = \frac{1}{H} \frac{\partial}{\partial \sigma} \left( \frac{\partial c^{n+1/3}}{\partial \sigma} + \frac{\partial c^n}{\partial \sigma} \right). \tag{7}$$

Note that the vertical exchange of sediment is with the adjacent layers. Additional boundary conditions at the water surface and sea bed are needed in order to obtain solutions.

3. Spatial discretization

3.1. Explicit Eulerian–Lagrangian method for advection

In the first fractional step, an explicit Eulerian–Lagrangian method is employed to approximate the advection term. Using the Taylor series expansion, concentration at a point, $p$, traced along a streamline over a time step $\Delta t$ can be approximated by the following expression:

$$c^n_p \approx c^n + \Delta t \left( u_j^n \frac{\partial c^n}{\partial x_j} \right). \tag{8}$$

Substituting Eq. (8) into Eq. (5) gives the following simple equation:

$$c^{n+1/3} = c^n_p. \tag{9}$$

This algebraic equation, Eq. (9), is independently calculated in each layer and at each node, so it can be easily vectorized for parallel computation.

3.2. Implicit FEM for horizontal diffusion

In the second fractional step, the standard Galerkin FEM is employed to implicitly discretize the horizontal diffusion equation, Eq. (6). The implicit discretized system for each layer is represented as follows:

$$\mathbf{D}c^{n+2/3} = \mathbf{M}c^{n+1/3}, \tag{10}$$

where $c = [c_1, c_2, \ldots, c_N]^{T}$, $\mathbf{D} = \mathbf{M} + \Delta t \mathbf{F}$; $\mathbf{M}$ denotes the integration of the products of the finite element isoparametric shape functions along the boundary in matrix form. If a constant horizontal eddy coefficient is used, matrix $\mathbf{D}$ becomes an invariant matrix for all layers. Thus, Eq. (10) can be simply written as

$$c^{n+2/3} = \mathbf{D}^{-1} \mathbf{M}c^{n+1/3}. \tag{11}$$

The inverse of the coefficient matrix, $\mathbf{D}^{-1}$, can be calculated in advance (before the time marching process) which leads to easy parallelization and very efficient computation.

3.3. Implicit FDM for vertical sediment exchange

The equation solved in the last fractional step, Eq. (7), is only concerned with the vertical direction thus can be independently solved at each water column. To achieve efficient computation, implicit FDM is employed to discretize this equation. A central finite difference scheme is used for all derivative terms in order to obtain second-order accuracy consistent with the horizontal discretization in the second fractional step. Since the vertical sediment exchange plays a very significant role in 3D transport computations, the applied scheme should maintain the mass conservation in each water column. The numerical discretization in a column must satisfy the following equality that is derived from the vertical integration of Eq. (7) over the water depth.

$$\int_0^1 \frac{\partial q_s}{\partial \sigma} \, d\sigma = q_{sb} - q_{ss},$$

where $q_s$ represents the vertical sediment exchange, $q_{sb}$ and $q_{ss}$ are the net sediment fluxes at the bottom and water surface, respectively. By using the following sediment exchange relationships at the seabed and water surface:

Seabed: 
$$q_{sb} = \left( \left. \left( \omega_a c + \frac{\partial c}{\partial \sigma} \right) \right|_{\sigma=0} \right) = \begin{cases} x_o \omega_b (c_1 - c_{ss}) & (\tau_b > \tau_b, cr), \\ x_o \omega_b c_1 & (\tau_b \leq \tau_b, cr), \end{cases}$$

Water surface: 
$$q_{ss} = \left. \left( \left. \frac{\partial c}{\partial \sigma} \right) \right|_{\sigma=1} \right) = 0$$

the finite difference expression for Eq. (7) can be derived as follows:
\( \mathbf{B} \mathbf{e}^{n+1} = \mathbf{F} \)  

in which, \( \alpha \) is a coefficient related to local sediment properties, \( \mathbf{e} = [e_1, e_2, \ldots, e_L]^T \), \( \mathbf{F} = [f_1^{n+2/3}, f_2^{n+2/3}, \ldots, f_L^{n+2/3}]^T \), \( \mathbf{B} \) is a triangle matrix with the following elements:

\[
B_{11} = 1 + \delta \lambda_2, \quad B_{12} = -\delta (\lambda_2 + \theta_2), \\
B_{21} = -\frac{1}{2} \delta \lambda_2, \quad B_{22} = 1 + \frac{1}{2} \delta (\lambda_3 + 2\lambda_2), \\
B_{23} = -\frac{1}{2} \delta (\theta_3 + \lambda_3 + \lambda_2), \\
B_{ll} = 1 + \frac{1}{2} \delta (\lambda_{l+1} + 2\lambda_l + \lambda_{l-1}), \\
B_{ll+1} = -\frac{1}{2} \delta (\theta_{l+1} + \lambda_{l+1} + \lambda_l), \quad (l = 3, \ldots, L - 2), \\
B_{l+1, l-2} = \frac{1}{2} \delta (\theta_{l-2} - \lambda_{l-1} - \lambda_{l-2}), \\
B_{l+1, l-1} = 1 + \frac{1}{2} \delta (2\lambda_{l-1} + \lambda_{l-2}), \\
B_{l-2, l} = -\frac{1}{2} \delta \lambda_{l-1}, \quad B_{l+1, l-1} = \delta (\theta_{l-1} - \lambda_{l-1}), \\
B_{ll} = 1 + \delta \lambda_{l-1},
\]

\( \delta = \frac{\Delta t}{\Delta x} \), \( \lambda_l = \left( \frac{\varepsilon_s}{H^2 \Delta \sigma} \right)_{il} \), \( \theta_l = \left( \frac{\alpha_b}{H} \right)_{il} \).

Eq. (12) can be efficiently solved using the double sweep method in each water column and is easily parallelized for parallel calculation.

### 4. Parallel algorithm

Parallel computational efficiency depends greatly on the decomposition strategy, communication architecture and parallel algorithm being employed. Domain decomposition strategy and master–slave communication architecture are implemented to parallelize the sequential model just described in the previous sections. A new parallel algebraic solver especially for solving non-symmetric matrices resulted from time variant irregular domains due to flooding and drying of land boundary has also been developed to improve the parallel computational efficiency. To demonstrate the accuracy and efficiency of the parallel model, two real applications had been conducted.

#### 4.1. Domain decomposition

The current approaches for parallelizing sequential models are domain decomposition and functional decomposition. The domain decomposition strategy is to decompose a domain of interest into a number of subdomains and assign the subdomains to different processors for parallel processing. The domain decomposition is probably the most widely used method for parallelizing numerical calculations because it introduces natural parallelism on parallel computers. It has been applied to problems across a wide variety of disciplines with generally good results (e.g. [2,9,22]).

Functional decomposition, called equation decomposition as well, partitions a sequential model into different functional parts. Each functional part is assigned to a processor and solved independently. Functional decomposition requires data communication in the whole domain of interest. In practice, using only functional decomposition for dependent systems may demand a large amount of data communication because the solutions are generally dependent between functions. Functional decomposition scheme is sometimes used in combination with domain decomposition for solving relatively separated systems (e.g. [21,33]). The functional decomposition method is not appropriate for partitioning the present model because all sub-step calculations depend on the previous sub-step results.

Domain decomposition strategies are popular in computational fluid dynamics applications. The basic idea is to decompose a spatial domain, where the problem is defined, into several subdomains and assign the subdomains to different processors for synchronous calculations. The correct physical relations between subdomains are maintained during the computation process by inter-processor communications. To minimize the overhead due to uneven amount of data transfer between processors, a domain decomposition scheme should be carefully constructed, especially for coastal and estuarine areas in which complicated geometries are commonly encountered. In general, it is desirable to decompose a physical domain into subdomains with simple and regular geometry for easy algorithm implementation [35]. Two widely used decomposition strategies are the (i) strip and (ii) patch decomposition. The strip decomposition scheme partitions a domain into a number of ‘strip’ subdomains and each subdomain only exchanges message with the two adjacent subdomains. The patch domain decomposition scheme partitions a domain into a number of subdomains in ‘patch’ form and each subdomain is required to communicate with the four neighboring subdomains. The communication time of patch decomposition reduces significantly with increasing number of processors being used [10,35], while the communication time of strip decomposition is invariant so it will not benefit from more processors.

Strip domain decomposition is suitable for partitioning narrow and long domains such as rivers and channels because the amount of data exchange in narrow interconnects are usually very small. Patch domain decomposition can efficiently reduce the communication time with increasing the number of processors. Hence,
choosing a domain decomposition scheme with minimal message exchange depends on the actual geometry of the area of interest. The domain decomposition strategy used in this parallel model is a mixed domain decomposition scheme which combines both the strip and patch domain decomposition.

Usually, areas of interest in estuarine and coastal regions have complicated irregular geometry. For instance, a domain often encountered in estuarine and coastal numerical modeling consists of narrow and long inflow rivers, islands and a wide estuary region. The strategy used here is to apply strip domain decomposition to narrow and long regions and patch domain decomposition to decompose wide areas thus the overall communication time can be shortened when large number of processors are being used.

Uneven workload (computational load and/or communication load), also known as local balancing problems, among processors will arise when numbers of grid points assigned to processors are not the same, or the computational time is not stable or unpredictable for a grid point processing. A processor, which has finished a computation earlier, must be idle to wait for instruction or message from its neighboring processors that are still in operation. This will consume CPU time and decrease the parallel processing efficiency. With the present numerical method, the amount of computation of each grid point is the same. Hence, the domain decomposition strategy implemented here will assign relatively same grid points per processor to balance the workload among processors.

4.2. Communication

Master–Slave (M–S) communication strategy is employed in the present numerical model for data transfer between adjacent processors. The M–S approach is very popular in CFD applications on MIMD parallel machines. In M–S communication, data transfer is only limited between a designated ‘master’ processor and the other active processors assigned as ‘slaves’. The master mainly plays the role as a coordinator and the slaves are the workers. The master distributes/scatters requisite data to the slaves for computations and gathers back the computed results from the slaves for outputs or further communications. More specifically in this model, the master is responsible for (i) reading the initial grid information, (ii) partitioning the prescribed problem, (iii) controlling the slave operations, (iv) communicating with the slaves and outputting the final solutions. The master can run either on an external remote front end machine, or on one of the processing elements of a MIMD array itself which could be dedicated to the spawning of data. The slaves are mainly responsible for (i) receiving data from the master, (ii) performing computations of the subdomains and (iii) sending the solutions back to the master. For implicit numerical schemes, the intermediate solutions of the interconnection nodes have to be interchanged with the solutions possessed by the neighboring slave processors. Thus, the slave processors send back all the intermediate solutions of the interconnections, which are necessary for subsequent computations by the other slaves, to the master. The master after gathering all the solutions will then scatter the necessary data to the relevant slaves for subsequent computing processes. The Message Passing Instruction (MPI), which is one of the standard parallel communication instruction languages [19], is adopted for the model’s M–S communication instructions so that the model can be easily ported on other parallel machines.

Heavy traffic of data flow may give rise to bottleneck circumstances in M–S communication. This circumstance can be mitigated by using additional sub-master processors each taking charge of a certain number of slave processors. This approach can be applied recursively and the recursive process is easily parallelized. Although the other communication strategies such as the neighbor–neighbor (N–N) communication approach will not have this bottleneck problem, a message in N–N approach may have to make several transits among processors before it can reach the destined processor which needs the message. This will increase the time spent on each communication operation, which consists of the start-up time and the data transfer time. In comparing the M–S communication time with the N–N communication time under strip/patch domain decomposition architecture, it can be shown that the M–S communication time is slightly less than the N–N time [8,24]. Also, the M–S strategy is more portable and easy to code with MPI.

4.2.1. Data structures for message passing

In FEM, the solution of a grid point depends on the values at the points immediately adjacent to it. For a subdomain, calculations at an interconnection point require the solutions at its adjacent points which are partly held by the neighboring processors. These adjacent points are called ‘halo points’ of the subdomain. Inside a subdomain, the values at the halo points are only for calculations and will not be updated. Therefore, each processor should be assigned with enough memory to store the halo point values that are updated in the neighboring processors. Fig. 1 shows the halo points around a subdomain with $4 \times 4$ nine-nodal quadrilateral elements. The points denoted by ‘$\times$’ and ‘$\circ$’ are the calculation points in the subdomain. The calculation points at the interconnections, symbolized by ‘$\circ$’, are the halo points of the neighboring subdomains, which values should be sent to the neighboring processors. The halo points in the subdomain are consisted of internal halo points denoted as ‘$+$’ and external halo points as

4.3. Matrix solver

The Jacobi and CG iteration methods are most widely used in parallel computations of the problems with implicit numerical schemes because these methods have no data dependency. However, the convergence rate of these iteration methods depends on the initial assumed values. Though preconditioned CG iteration [18] can greatly improve the convergence rate, the procedures are difficult to be parallelized. For a non-symmetric linear algebraic system, such as \( G^c = Z \), which is derived from the fully implicit finite-element continuity equation of the present model (see [17]), can be partitioned into \( p \) sets of algebraic subsystem as follows:

\[
G^c = Z \Rightarrow \begin{bmatrix}
G_{i1} & \cdots & G_{ip}
\end{bmatrix} = \begin{bmatrix}
Z_1 \\
\vdots \\
Z_p
\end{bmatrix}.
\]

Here, \( G_i = \{n_x \times (n_x + n_h)\} \), is a coefficient matrix held by the processor \( i \); \( \zeta_i = [n_x + n_h] \) is an unknown vector held by the processor \( i \); \( Z_i = [n_x] \) is a source vector held by the processor \( i \); \( n_x = m^2/p \), is the number of the calculation points in the subdomain \( i \); \( n_h = 4(m/\sqrt{p} + 1) \) is the number of internal halo points in the subdomain \( i \). Because the nodes in each processor are renumbered (re-ordered) as discussed earlier, the coefficient matrices in the subdomains have the same form and are consisted of two parts, i.e.

\[
G_i = \{G_{i,j} \mid G_{h,j}\},
\]

where \( G_{i,j} = \{n_x \times n_x\} \), is a coefficient matrix corresponding to the calculation points, \( G_{h,j} = \{n_x \times n_h\} \), is a coefficient matrix corresponding to the internal halo points. Similarly, the unknown vector \( \zeta_i \) is partitioned into two parts such as

\[
\zeta_i = \begin{bmatrix}
\zeta_{i,j} \\
\zeta_{i,h}
\end{bmatrix}
\]

in which \( \zeta_{i,j} = [n_x] \), is an unknown vector corresponding to the calculation points; \( \zeta_{i,h} = [n_h] \) is an unknown vector corresponding to the internal halo points. Therefore, the algebraic system of Eq. (13), is rewritten as

\[
G_{i,j}\zeta_{i,j} + G_{h,j}\zeta_{h,j} = Z_i.
\]

By using the Gauss elimination method to diagonalize the coefficient matrix \( G_{i,j} \), the following equation can be obtained:

\[
\zeta_{i,j} = Z_i' - G_{h,j}'\zeta_{h,j}.
\]

where \( Z_i' = G_{i,j}'Z_i \) and \( G_{h,j}' = G_{i,j}'G_{h,j} \), both are known coefficient vector and matrix, respectively. Eq. (17) in-

Fig. 1. Definition of calculation points and halo points in subdomains.

‘\( \odot \)’. All values at the halo points are obtained and updated in the neighboring processors. The values at the internal halo points are required for general calculations related to the finite element solutions, such as the calculations performed by the horizontal diffusion solver, continuity equation solver, etc. The values at the external halo points are for the Eulerian–Lagrangian computation (streamline tracing) in the advection solver. The existence of the external halo points depends on the strength of flows. If the flow velocity is small and the streamlines are within the internal ‘halo’ area, the external halo points will not be needed. To ensure efficient message passing without broken arrays, the node numbers are re-ordered starting from the interior points of the subdomain, to the interconnection points, to the internal halo points and finally to the external halo points in a contiguous array.
method depends on the number of grids held by each processor. The computational efficiency is higher for the GJ method than the Jacobi method when the number of grids handled by each processor are between 4 and 2000. When each processor holds about 40–100 calculation points, the GJ method reaches the uppermost efficiency that the computational amount is only one-tenth of the Jacobi method. Too many or too few grids held by a processor can cause computational inefficiency for the GJ method. This is because the Gauss elimination process will be timely when many calculation points are involved and the Jacobi iteration’s convergence rate will be slow when too many halo points are held by a subdomain.

The amount of message exchange in one iteration is almost the same for the two methods. However, if the number of iterations in the GJ method is much less than the Jacobi method, the total amount of message exchange is also reduced. This results in a better computational efficiency.

On the whole, the GJ iteration method performs better than the Jacobi method in both areas: higher parallel computational efficiency and less communication loading. Furthermore, the computational efficiency of the GJ method may be even more beneficial from estuarine and coastal applications because the irregular coastline geometries imply less number of halo points than the maximum number presented in the above example.

5. Applications

The parallelization of the 3D numerical model constructed with the HOS discretization method is achieved by implementing the strip/patch domain decomposition strategy, M–S communication scheme and the new GJ algebraic system solver. The parallel model is coded with the standard computer language, FORTRAN 90, to perform the master–slave scheme. The model is ported
on an MIMD structured parallel computer, IBM SP2/900, which is consisted of two frames, each of which has 16 processors. The flow chart of the parallelized model is shown in Fig. 3. The initialization step for communication is first conducted to determine which processor is to be used as the master and how many slave processors are available. The communication between the master and the slaves is connected by the wide arrows in Fig. 3. With MIMD architecture, the master reads the initial grid and boundary information and decomposes the global domain into a number of subdomains, same as the number of available slaves, with the domain decomposition method as described. The subdomain information is sent to each slave to allocate memory for receiving subdomain grid information. The slaves renumber the nodes for efficient communication. After the parallel system is established, the master interpolates the boundary conditions, assists the slaves’ communication and outputs the computed results, while the slaves carry out the computations on their assigned subdomains. Communication between the master and the slaves is required during the time marching process. The flow chart and communication in the time marching process are shown in Fig. 4. Though the master has idle time waiting for messages from the slaves, the CPUs of the slaves are in full operation during the time marching.

5.1. Application to Rusi Bay

5.1.1. General description

Rusi Bay is located in Jiangsu Province, China, north of Yangtze River at the latitude 32.2°N and longitude 120.3°E (see Fig. 5). The bay has a surface area of about 1080 km² with an East–West length of 53 km and an opening 28 km wide. A deep waterway, called Xiao Miao Hong channel, is located in the center of the bay. Beside the channel, there are two wide shoals which dry up at low water level and is flooded at high water level.

![Fig. 3. Flow chart of the 3D parallelized hydrodynamic and sediment transport model.](image-url)
In a typical tidal condition, about 40% of the bay will dry up as the water elevation reaches the lowest level. Two submerged bed dunes scatter over the opening of the bay and partition Xiao Miao Hong channel into three branches, i.e. North, middle and South branches. The tidal difference averaged in spring tides is about 4.3 m while the annual mean tidal difference is about 2.7 m. The large tidal difference induces very strong tidal currents in the bay. The maximum observed velocity is over 2 m/s.

The main source of suspended sediments in the bay is from local resuspension. The sediment source from the sea is very small and can be neglected according to the analysis of the remote sensing data [15]. According to field measurements, the bed sediments in the bay are from coarse sand to fine clay corresponding to the particle sizes from 0.004 to 0.24 mm. The bottom sediments in the shoals having median particle size of 0.12 mm are coarser than that in the channel bed. However, the median suspended particle size is 0.08 mm, finer than the median bed sands. This implies that the energy generated from normal hydrodynamic forcing may not be able to entrain the coarser bottom particles. But the

---

**Fig. 4.** Flow chart showing the computation and communication processes in time marching.

**Fig. 5.** Map of Rusi Bay and the locations of the tide stations.
particles can be resuspended in storm surge conditions. The bay is basically in a dynamic equilibrium erosion–deposition situation according to the historical bathymetric data [15].

5.1.2. Model establishment and verification

The field data available for the model calibration were measured by Nanjing Hydraulic Research Institute in September 1992. The tidal fluctuations were observed at the Rusi fixed tide gauge and the tidal currents and sediment concentrations were simultaneously measured at the five tide stations shown in Fig. 5. The water column is divided into six layers for the whole region. The grid arrangement with the nine-nodal quadrilateral finite elements consisting of 631 nodes and 135 finite elements in each horizontal layer is shown in Fig. 6. The tidal elevations are imposed at the open boundary. The sediment physical parameters used in the present simulation are the same as those employed in the 3D multi-layer simulation conducted by Wai et al. [30].

For parallel computations, the global domain (the whole area of interest) is partitioned into eight subdomains with the domain decomposition method. Each subdomain receives the same number of calculation points to avoid uneven workload between processors in parallel processing. Fig. 6 also shows the partitioned subdomains used in the simulation. Parallel computation is carried out with nine processors, one of the processor is used as the master and remaining eight are slaves responsible for the computations for the eight subdomains. The simulation period is one day in a spring tide condition (24.8 h) in which field measurements are available. A large time step of 600 s is employed in the simulation in order to demonstrate the model stability.

As mentioned, about 40% of the bay area will dry up in ebb tides. These flooding and ebbing processes in a wide floodplain greatly affect the flow pattern. During flood tides, the tidal currents from the sea start to enter the bay through the deep channel and then diverge to the shallower areas. Thus the flow velocity in the deep channel decreases due to the divergence of the channel flows. In opposite, the water stored in the shallower areas merges into the deep channel during ebb tides leading to an increase of flow velocity in the channel. To accurately account for these processes, the ‘drying up’ technique developed by Lu and Wai [16] has been employed.

Fig. 7 shows the comparison between the computed tidal elevation and the measured elevation at the Rusi fixed tide gauge. The computed tidal elevations agree quite well with the observations. To illustrate the tidal flooding and ebbing process, the flow pattern at each layer during a tidal flood (at the 6th h) and a tidal ebb (at the 14th h) are plotted in Figs. 8 and 9, respectively. Fig. 8 shows the flooding velocity vectors which cover the entire bay indicating the area is completely inundated. Small swells are observed at all levels near the center of the south boundary. In general, the velocity intensity is fairly uniform throughout the bay. Fig. 9 shows that the entire floodplain has dried up at 14 h when the water elevation reaches almost the minimum and the ebb flows mainly align in the channel region.

The verification of suspended sediment concentration in Rusi Bay is difficult because of the noise in the observations. To reduce the noise in the observations, the vertical averaged concentrations are used. The time series of the computed and observed vertical averaged concentrations at the four observing stations are plotted together in Fig. 10 for comparisons. The computed vertical averaged concentrations generally agree with the observed values. The tidal averaged concentration profiles, which averaged within a flood or ebb period, are used to demonstrate the computational accuracy in the water column. Fig. 11 shows the comparisons of the computed tidal averaged concentration profiles against the observed profiles. It can be seen that the predicted concentration profiles concur closely with observed profiles.
5.2. Application to the Pearl River estuary

5.2.1. General description

The Pearl River estuary (PRE) is located in Guangdong province and Hong Kong, China between the longitudes of 113°33' and 114°09' and latitudes of 22°12' and 22°45'. The estuary looks like a tilted bell with open boundary dimensions of about 4 km freshwater inlet in the north and 60 km seaward opening in the south (between Hong Kong and Macau.) The longitudinal length of PRE is about 63 km from the inlet to the seaward opening (see Fig. 12). There are four major river inlets carrying freshwater and sediments into the estuary, and they are: Humen in the northern side and Jiaomen, Hongqilimen and Hengmen in the western side. Beginning from Humen, there are two natural waterways divided by Lantau Island, called the West Channel and the East Channel, which are the main navigation channels in PRE. Three large shoals distributing in between the two waterways are: the West, Mid and East shoals and among which the West shoal is the largest and shallowest. Hong Kong waters is located in the southeast of PRE and is comprised of several deep commercial channels, such as Urmston Channel, Lamma Channel, Victoria Channel, etc. Lot of islands scatter in Hong Kong waters, including Lantau Island, Hong Kong Island, Lamma Islands, etc., that leads to a very complex hydrodynamic characteristic in this area. The area of interest for this application consists of Hong Kong waters and part of the South China Sea.

5.2.2. Model establishment and verification

Since there is no accurate field observed sediment information available for the entire PRE for verification, the parallel application is focused on the simulation of the hydrodynamic patterns in the estuary. For the model area, the northern open boundaries are set at the en-
trances of the four river inlets, which supply the upstream run-off and sediments as mentioned. The sea boundary is laid down by linking the islands as shown in Fig. 12. The water column is divided into six layers vertically and each horizontal layer is partitioned into 135 nine-nodal quadrilateral finite elements with 633 nodal points. Similar to the Rusi Bay application, nine processors are used to perform the parallel computations. One processor is used as the Master to read model information and control the computations. Eight processors are assigned as the slaves to be responsible for carrying out the subdomain computations. The domain of interest (the model area) is decomposed into eight subdomains by the combined strip/patch domain decomposition scheme as earlier and the decomposed subdomains are plotted in Fig. 13. The combined scheme reduces the number of halo points and this will shorten the communication time and the GJ iteration process. To achieve a well local balanced parallel computation, each subdomain consistently contains relatively the same number of calculation points, which is between 79 and 80.

The field data available for model calibrations and result verification are provided by the Hong Kong Government Civil Engineering Department (HKCED) for model calibrations and result verification. The field data available for model calibrations and result verification are provided by the Hong Kong Government Civil Engineering Department (HKCED) for model calibrations and result verification.
measured in the wet season of July 1990 during a spring tidal cycle. The locations of the tide gauges and tide stations are shown in Fig. 12. The HKCED field survey was synchronously performed at nine tide gauges and seven tide stations in Hong Kong waters (E1–E6 and EW). There are seven open boundaries in the application. Based on the available field data, the Southern and Eastern open boundaries are controlled by tidal elevations while the northern open boundaries at the four river entrances are governed by water discharges.

The prescribed boundary tidal elevations are obtained by extrapolating the observed tidal elevation information at the nearest tide gauge to the boundaries according to the wave propagation theory [11]. The tidal wave propagating speed, known as \( gh \), is used to calculate the phase lag between tidal waves. Thus, the boundary tidal elevations can be estimated based on the information of the local water depth and the known tidal elevations at the selected tide gauges. Besides the phase lag, the shape (amplitude and wavelength) of a tidal wave will be deformed (refracted) during propagating into the shallow water zone. The tide amplitude increases when it propagates from the sea (deep water) to the estuary (shallow water). This method only provides a quick estimation for the boundary water elevation without considering the bottom friction. Hence, boundary conditions obtained by this method have to be calibrated for the best possible final result.

Fig. 14 shows the comparisons of the computed tidal elevations with the observed tidal elevations at seven tide stations. The computed tidal elevations at North Point and West Lamma agree quite well with the observed tidal elevations. At the Macau and Soko Island tide gauges, the computed tidal elevations generally coincide well with the observed time series with an acceptable 2% averaged relative error in Macau during the flood and Soko Island during the ebb. The averaged relative error at and Tap Shek Kok is around 3%.

The comparisons between the observed and computed vertical averaged velocity time series are shown in Fig. 15 at station E1 and Fig. 16 at station EW. In the figures, the square solid dot represents the observed vertical averaged velocity and the error bar indicate the fluctuation range of the observations in the water column. The solid line depicts the computed vertical averaged velocity in time series. The computed time series are generally in good agreement with the observed values except at the time of the second ebb tide. Good comparisons are found at the stations which are far
away from the open boundaries such as station EW. Larger errors are found at the time during tidal flooding at E1, which may be due to the open boundary effect. This suggests that the model boundaries should not be set too close to the area of interest. Figs. 17 and 18 show the tidal averaged velocity profiles (averaged in two flood tides and two ebb tides) at stations E1 and EW, respectively. The computed profiles basically correlate well with the observed profiles at these two stations. The water surface flow patterns during a flood tide and an ebb tide are shown in Figs. 19 and 20, respectively. Fig. 19 demonstrates that when the incoming tidal wave reaches Humen, the flow in the Eastern boundary starts to turn to ebb while most parts of Hong Kong waters are still in slack water condition. The flow pattern depicted in Fig. 20, was generated during the second ebb tide in which the flow is stronger than the first ebb. It is noticed that the major flow pattern primarily aligns with the two natural waterways in the estuary and the eastern part of Hong Kong waters is already experiencing the beginning of a flood tide. The computed patterns are essentially conformed to the flow patterns observed in the estuary [32].
6. Computational speed-up

To demonstrate the computational speed-up of the present model with the proposed parallelization algorithm, the PRE simulation has been re-run using the conventional sequential computer code. The results computed by the two computational approaches (parallel and sequential) are thoroughly compared at every time step and nodal point. In theory, there should be no differences between the values computed by either the parallel approach or the sequential approach. However, the results may not be identical if the calculations are carried out in different computer systems because of the differences between computer systems in treating truncation errors. In comparing the computed results of the two approaches, the maximum relative error for the surface water elevation is found to be $10^{-8}$ and for flow velocities and sediment concentrations is $10^{-6}$. These errors resulting from the truncation errors are not significant and acceptable for hydrodynamic and sediment transport studies.

By using the domain decomposition strategy, the computed domain, which consists of islands, river inlets and complicated coastline, is partitioned into two, four, eight and sixteen subdomains for the speed-up investigation, respectively. The simulation time is a one-day tidal cycle, approximate 24.8 h covering two complete tidal cycles. A simulation using the sequential code is performed in one processor to acquire the reference CPU time. Therefore, the speed-up for the parallel computation, $S_p$, can be expressed as the division of the sequential CPU time, $T_s$, by the parallel CPU time, $T_p$, as suggested by Bender et al. [1], such as

$$S_p = \frac{T_s}{T_p}.$$  \hfill (20)

Fig. 21 shows the speed-up of the present parallelized model with the number of slaves. The CPU times and speed-ups for the PRE simulation are listed in Table 1. As indicated in the figure and table, the parallelized model increases the computation speed sharply with more processors. The figure reveals that the parallel model can speed-up the computation about five times per additional processor when the number of processors being used is less than five, and the speed-up rate slows down when the number of processors exits five. This is primarily due to the large ratio of the halo points to the calculation points, $R$, and this will decrease the efficiency in the GJ iteration process.

Theoretically, the speed-up for parallel computation increases one time per additional processor if the same iteration method is employed for solving algebraic equations in both the sequential and parallel models. The much larger speed-up achieved by using the present parallel model is because of the improvement in parallel computation by using the fully parallelizable GJ iteration method.

The speed-up ($S_p$), due to model parallelization, cannot well indicate the performance of an individual processor when handling a particular size problem.

![Fig. 20. Surface flow pattern of the PRE simulation during an ebbing period.](image)

![Fig. 21. Plots of speed-up and coefficient of performance (COP) of the parallelized model with number of slaves.](image)

Table 1: The CPU time, speed-up and coefficient of performance (COP) of the parallel computations in the PRE application

<table>
<thead>
<tr>
<th>Number of slave processors</th>
<th>CPU time (s)</th>
<th>Speed-up</th>
<th>COP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
</tr>
<tr>
<td>Sequential</td>
<td>8391</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>1763</td>
<td>4.76</td>
<td>2.38</td>
</tr>
<tr>
<td>4</td>
<td>579</td>
<td>14.49</td>
<td>3.62</td>
</tr>
<tr>
<td>8</td>
<td>340</td>
<td>24.68</td>
<td>3.08</td>
</tr>
<tr>
<td>16</td>
<td>250</td>
<td>33.56</td>
<td>2.10</td>
</tr>
</tbody>
</table>
Thus, another indicator is introduced to describe the parallelization performance of a model. The coefficient of performance (COP) for model parallelization is defined as the values of speed-up divided by the number of processors, i.e.

$$E_p = \frac{S_p}{p}$$

(21)

The COP of the model parallelization for the PRE application is also plotted in Fig. 21. The figure reveals that the performance reaches the maximum when five processors are being used and the maximum COP is about 3.7 (370%). The number of processors being used in achieving the maximum performance is known as ‘critical processor number’ and the critical processor number is five for the PRE application. In general, the maximum COP and the critical processor number depend on the problem size. For a given problem size, the critical processor number can be estimated by the calculation points being held by each processor. The optimum node number is about 40–100 nodes per subdomain as mentioned earlier. Again, the 3.7 COP achieved in the present parallel model is attributed to the computation improvement by using the GJ parallel iteration method.

7. Conclusions

It has been demonstrated how a parallel processing algorithm in addition with a new parallel algebraic solver can be effectively incorporated into an efficient numerical scheme to significantly enhance the computational performance of a coastal transport simulation.

The mixed domain decomposition strategy, which is a combination of the strip and the patch domain decomposition methods, is used to partition the complicated domains often encountered in estuarine and coastal simulations. The master–slave communication system and the node renumbering scheme implemented in each subdomain are for the improvement of the communication efficiency. The new GJ iteration method, which is based on the Jacobi iteration method and the initial estimate by using the local Gauss elimination method, is developed for parallely solving the linear algebraic systems in coastal transport applications. The GJ method greatly increases the computational efficiency, reduces the number of iterations and message communication.

The accuracy of the model has been examined in two real-life large-scale applications in which the computed results agree reasonably well with the available field observations. The implementation of the parallel processing algorithm and the GJ iteration method improves the computational speed by fivefold per additional processor and the model parallelization performance close to 370%. The parallel model will allow more coastal processes parameters to be investigated and timely simulations to be conducted. This will broaden our physical insights of the coastal transport processes as well as the model’s engineering relevance.

Acknowledgements

This work was supported in part by the Central Grant B-Q086 of The Hong Kong Polytechnic University. Additional support was received from the Hong Kong Research Grants Council under the grant number: PolyU 5038/98E.

References