A high-order harmonic polynomial method for solving the Laplace equation with complex boundaries and its application to free-surface flows. Part I: two-dimensional cases.

J. Wang[1,*], O. M. Faltinsen[2] and W. Y. Duan[1]

[1]*Institute of Marine Hydrodynamics, Harbin Engineering University, Harbin, China*

[2]*Centre for Autonomous Marine Operations and Systems (AMOS), Norwegian University of Science and Technology, N-7491 Trondheim, Norway*

A high-order Harmonic Polynomial Method (HPM) is developed for solving the Laplace equation with complex boundaries. The "irregular cell" is proposed for the accurate discretization of the Laplace equation, where it is difficult to construct a high-quality stencil. An advanced discretization scheme is also developed for the accurate evaluation of the normal derivative of potential functions on complex boundaries. Thanks to the irregular cell and the discretization scheme for the normal derivative of the potential functions, the present method can avoid the drawback of distorted stencils, i.e., the possible numerical inaccuracy/instability. Furthermore, it can involve stationary or moving bodies on the Cartesian grid in an accurate and simple way. With the proper free-surface tracking methods, the harmonic polynomial method has been successfully applied to the accurate and stable modeling of highly-nonlinear free-surface potential flows with and without moving bodies, i.e., sloshing, water entry and plunging breaker.

1. Introduction

The Laplace equation is important in many fields of science, notably the fields of fluid dynamics, electromagnetism and astronomy. Particularly in fluid dynamics, the Laplace equation is widely used to describe water waves [1, 2] and wave-body interaction [3, 4]; and the flow governed by the Laplace equation is known as potential flow. Furthermore, the Laplace equation corresponds to the general solution of Poisson equations, resulting in potential applications to the viscous flows governed by the Navier-Stokes equations [5, 6]. For the solution of the Laplace equation with complex boundaries, numerical methods are widely used. The reason is simply that it is commonly difficult to seek the analytical solution with the presence of complex boundaries. In marine hydrodynamics, the numerical prediction of the potential flows with free-surface boundaries is highly interesting. It is applied to sloshing[7], propagating waves [1, 2], wave resistance of travelling ships [8], wave-body interactions [3, 4]. The accurate and stable modeling of these phenomena can be numerically challenging, particularly when stationary/moving bodies are present and the nonlinear effect matters [9, 12]. The efficiency of numerical methods is also an important aspect, especially for large-scale problems.

The main approaches for the free-surface potential flows can be classified into four categories: the methods that solve model equations such as Boussinesq type equations [11], the high-order spectral methods[12], the Boundary Element Methods (BEM) and the field methods such as Finite Difference Methods (FDM)[13-15] and Finite Element Methods (FEM)[16, 17]. The first two methods are widely used in water waves. To involve stationary/moving bodies, the boundary element methods and the field methods are widely used. BEMs solve a boundary integral equation which is a consequence of applying Green's 2nd identity to the Laplace equation. The BEMs have two main advantages: they can describe highly-nonlinear waves accurately; the bodies are involved in a straightforward way, i.e., as boundaries. A conventional BEM requires $O(N^2)$ operations to discretize the boundary integral equation, resulting in

* Email address for correspondence: jingbowang@hrbeu.edu.cn

a dense matrix. Here N is the number of unknowns on the boundaries of the computational domain. To solve the dense matrix system, a direct method such as the Gaussian elimination or LU-factorization commonly takes $O(N^3)$ operations and an efficient iterative solver could take $O(N^2)$ operations. Because of the computational complexity of at least $O(N^2)$, it is expensive for large-scale problems. It was reported that it is possible to reduce the computational complexity to O(N) by the fast multipole accelerated (FMA) methods [18, 19]. The field methods discretize the whole computational domain and operate with sparse matrices. The field methods can be more efficient than the conventional BEM for large-scale fully-nonlinear problems [20]. Towards accurate and efficient fully-nonlinear potential-flow solvers, a novel field method, called Harmonic Polynomial Cell method, was proposed by Shao & Faltinsen a few years ago [21, 26]. In the two-dimensional space, the method constructs a cell, involving four neighboring elements on structured grids, to discretize the Laplace equation by the high-order harmonic polynomials. In the three-dimensional space, the harmonic polynomial cell involves eight neighboring elements on structured grids. Their comparative studies showed that the harmonic polynomial cell method is much more efficient than the conventional BEM and the fast multipole accelerated BEM (FMA-BEM). Because the harmonic polynomial cell method uses the high-order harmonic polynomials, it can be much more accurate than the BEMs and other low-order field methods[21, 26]. The two-dimensional harmonic polynomial cell method has been investigated in depth by Ma et al. [23] and advanced by Hanssen et al. [24]. The harmonic polynomial cell method works very well on standard or slightly distorted cells; the accuracy can become poor on distorted cells [23]. These will be clearly shown in Section 3. The poor accuracy, on the distorted cells often encountered by highly-deformed free surfaces or complicate body boundaries, can result in numerical instability or even destroy the numerical solver. To overcome this drawback, Hanssen et al. [24] presented two alternative strategies, i.e., the immersed boundary method and the multigrid method. These strategies can improve the accuracy and stability for fully-nonlinear free-surface flows. However, they can also introduce new challenges: spurious force oscillations may arise for moving bodies when using the immersed boundary method [25, 29]; the generation of high-quality boundary-fitted grids, used in the multigrid method, can be a cumbersome and time-consuming task especially for the three dimensional cases with complex boundaries. In this paper, we will propose the 'irregular cell', which avoids the drawback of the distorted harmonic polynomial cell, to deal with complex boundaries. We also developed an advanced high-order numerical scheme for the normal derivative of potential functions on boundaries. The high-order scheme works stably in the practical simulations of highly-nonlinear free-surface flows with and without moving bodies. Further, we present a locally-refined Cartesian grid system, which can be efficient without the loss of accuracy, to discretize the computational domain.

The paper is organized as follows: Section 2 presents the Laplace equation with boundary conditions and the fully-nonlinear free-surface conditions; Section 3 details the irregular cell (with comparison to the harmonic polynomial cell) and the new discretization scheme for the normal derivative of the potential functions on boundaries; Section 4 describes the locally-refined Cartesian grid system for the discretization of the computational domain; Section 5 outlines the methods for tracking the evolution of free surfaces; Section 6 gives some numerical examples of highly-nonlinear free-surface flows with and without moving bodies to verify and validate the proposed method. Section 7 summarizes the work.

2. Governing equations

We consider the Laplace equation in the domain Ω with either Dirichlet or Neumann conditions prescribed over different and exclusive portions of the domain boundaries Γ_D and Γ_N respectively:

$$\nabla^2\varphi(\mathbf{x}) = 0, \text{ in } \Omega \tag{1.a}$$

$$\varphi(\mathbf{x}) = f_D(\mathbf{x}), \text{ on } \Gamma_D \tag{1.b}$$

$$\frac{\partial\varphi(\mathbf{x})}{\partial n} = f_N(\mathbf{x}), \text{ on } \Gamma_N \tag{1.c}$$

where $\mathbf{x}$ denotes the position vector of field points and $\partial\Omega = \Gamma_D \cup \Gamma_N$ is the whole boundary of the domain Ω.

In the potential-flow theory, φ represents the velocity potential, giving the velocity of fluid particles by $\mathbf{u}(\mathbf{x}) = \nabla\varphi(\mathbf{x})$. For the water flow with free surfaces, the velocity potential over the water domain will be found by the solution corresponding to the following boundary conditions: the Dirichlet boundary condition, (1.b), is commonly used on free surfaces, S_F; the Neumann boundary condition, (1.c), represents the impermeability boundary condition over body boundaries with $f_N(\mathbf{x})$ equal to the normal velocity of the body boundaries. The evolution of the free surface is governed by the fully-nonlinear kinematic and dynamic boundary conditions

$$\frac{D\mathbf{x}}{Dt} = \nabla\varphi, \text{ on } S_F \tag{2}$$

$$\frac{D\varphi}{Dt} = \frac{1}{2}|\nabla\varphi|^2 - g\zeta, \text{on } S_F \tag{3}$$

where ζ is the free-surface elevation and the operator, $\frac{D}{Dt} := \frac{\partial}{\partial t} + \nabla\varphi \cdot \nabla$, is the substantial derivative following the water particle on the free surface. Eq. (3), derived from the Bernoulli's equation, is simply that the water pressure is equal to the constant atmospheric pressure on the free surface.

3. Harmonic-polynomial discretization schemes

The solutions of the Laplace equation are called harmonic functions. Given a two-dimensional harmonic function $\varphi(x,y)$, there exists a 'conjugated' harmonic function $\psi(x,y)$, satisfying the Cauchy-Riemann equations: $\psi_x = -\varphi_y$ and $\psi_y = \varphi_x$. The resulting pair of the harmonic functions can construct a complex analytic function, $f(z) = \varphi(x,y) + i\,\psi(x,y)$, where $z = x + iy$ with $i = \sqrt{-1}$. In potential-flow theories, $\varphi(x,y)$ is often called the potential function and $\psi(x,y)$ the stream function. Provided that $f(z)$ is regular near $z = 0$, we can expand it in a power series locally,

$$f(z) = \sum_{n=0}^{\infty} C_n z^n, \tag{4}$$

where $C_n = A_n + iB_n$. Representation (4) is mathematically complete. The complex polynomials, z^n, are analytic, resulting in the harmonic polynomials as follows

$$h_1 = 1, \tag{5.a}$$

$$h_{2n} = r^n \cos n\theta \ \ (n \geq 1), \tag{5.b}$$

$$h_{2n+1} = r^n \sin n\theta \ \ (n \geq 1). \tag{5.c}$$

Here, r and θ are the polar coordinates, satisfying $x = r\cos\theta$ and $y = r\sin\theta$. The harmonic polynomials, corresponding to the real and imaginary parts of z^n, satisfy the Laplace equation. Eqs. (5.b) and (5.c) can be written in Cartesian form, for example,

$$n = 1: \quad h_{2n} = x, \ h_{2n+1} = y \tag{6}$$

$$n = 2: \quad h_{2n} = x^2 - y^2, \ h_{2n+1} = 2xy \tag{7}$$

$$n = 3: \quad h_{2n} = x^3 - 3xy^2, \ h_{2n+1} = 3x^2y - y^3 \tag{8}$$

$$n = 4: \quad h_{2n} = x^4 - 6x^2y^2 + y^4, \ h_{2n+1} = 4x^3y - 4xy^3 \tag{9}$$

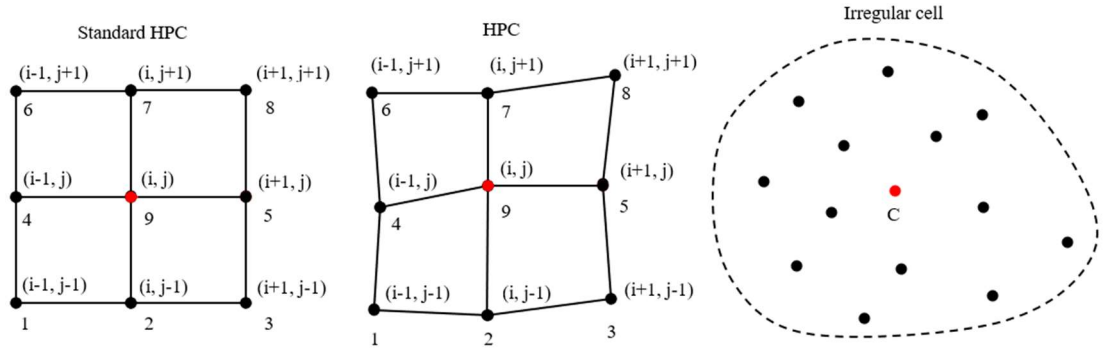


Fig. 1. Several cells used for discretizing the Laplace equation by harmonic polynomials.

Based on structured grids, the Harmonic Polynomial Cell (HPC) method has been proposed for discretizing the Laplace equation [21-24]. The HPC adopts 9 grid points in 4 neighboring quadrilateral elements, which is illustrated in Fig. 1. The standard HPC refers to the special harmonic polynomial cell, which consists of four square elements. The HPC uses the eight lowest-order harmonic polynomials to approximate the harmonic function $\varphi(x,y)$ in the vicinity of point 9, i.e.,

$$\varphi(x,y) = \sum_{j=1}^{8} b_j h_j(x,y). \tag{10}$$

Here the origin of the local Cartesian coordinate system located at point 9. The unknown coefficients b_j $(j = 1,2,\dots,8)$ can be found as the linear combination of φ_i $(i = 1,2,\dots,8)$ as

$$b_j = \sum_{i=1}^{8} c_{j,i}\varphi_i. \tag{11}$$

Here $c_{j,i}$ $(j,i = 1,2,\dots,8)$ is the elements of the inverse of the matrix D, whose elements are $d_{i,j} = h_j(x_i,y_i)$. From Eq. (10), the discretized Laplace equation at point 9 can be immediately expressed as $\varphi_9 = \varphi(0,0) = b_1 = \sum_{i=1}^{8} c_{1,i}\varphi_i$ or simply

$$-\varphi_9 + \sum_{i=1}^{8} c_{1,i}\varphi_i = 0. \tag{12}$$

Substituting Eq. (11) into Eq. (10) results in

$$\varphi(x,y) = \sum_{i=1}^{8}[\sum_{j=1}^{8} c_{j,i}h_j(x,y)]\,\varphi_i. \tag{13}$$

Over the harmonic polynomial cell, the gradient of the harmonic function can be evaluated by

$$\nabla\varphi(x,y) = \sum_{i=1}^{8}[\sum_{j=1}^{8} c_{j,i}\nabla h_j(x,y)]\,\varphi_i, \tag{14}$$

which was used to conduct the Neumann boundary condition

$$\partial\varphi/\partial n = \sum_{i=1}^{8}[\sum_{j=1}^{8} c_{j,i}\nabla h_j(x,y)\cdot\mathbf{n}]\,\varphi_i. \tag{15}$$

Here, $\mathbf{n}$ is the normal vector on the boundary. The harmonic polynomial cell was generalized to the three-dimensional space by using 27 grid points in 8 neighboring hexahedrons [26].

Eq. (12) represents the discretized Laplace equation inside the computational domain and Eq. (15) the discretized Neumann boundary condition on the boundary. Together with the Dirichlet boundary condition, i.e., Eq. (1.c), they are used to build up the global matrix system for the solution of $\varphi(x,y)$. It notes that the coefficient of φ_9 in Eq. (12) appears as a diagonal element in the global matrix system. Table 1 presents the coefficients of Eqs. (12) and (15) for the standard harmonic polynomial cell as indicated in Fig. 2. Compared to the off-diagonal coefficients, the diagonal coefficient has an opposite sign. Moreover, the absolute value of the diagonal coefficient is much larger than that of the off-diagonal coefficients, i.e., the resulting coefficients are diagonally dominant. These properties are advantageous to the stability of the numerical solution. Because of curved boundaries in practical problems, non-standard or distorted cells are commonly encountered. Distorted cells may reduce accuracy or even induce numerical instability. For example, the distorted harmonic polynomial cell in Fig. 2 could be a low-accuracy representation of the discretized Laplace equation at point 9, which will be shown later. A

proper harmonic polynomial cell could be the cell 2-3-10-5-7-6-11-4-2. Similarly, the cell 1-2-3-9-6-11-12-13-1 is better for discretizing the Neumann boundary condition at point 2. However, constructing a proper harmonic polynomial cell is not straightforward and could be very difficult near complex boundaries. To improve the quality of HPCs near free-surfaces, which is sensitive to the accuracy and stability of numerical schemes, Hanssen et al. introduced an additional boundary-fitted grid [24]. Their method gave better numerical results. For more complex boundaries, such as the overturning waves, the method encountered numerical difficulties (private communication). In the following, we will introduce the concept of 'irregular cell' for discretizing the Laplace equation and an advanced numerical scheme for the Neumann boundary condition.

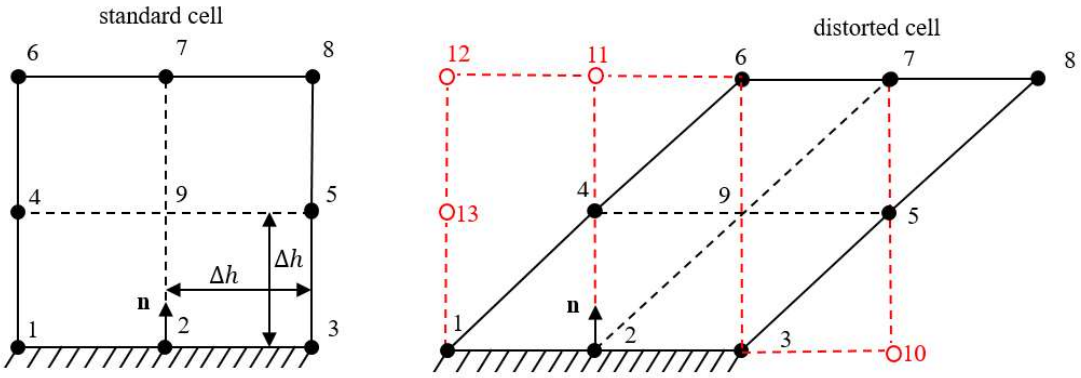


Fig. 2. Standard cell and distorted cells, used to discretize the Laplace equation at point 9 and the Neumann boundary condition at point 2.

Table 1. Matrix coefficients of the discretized Neumann boundary condition (at point 2) and the discretized Laplace equation (at point 9) for the standard harmonic polynomial cell shown in Fig. 2. The discretized equations are represented as $\sum_{i=1}^{9} \alpha_i \varphi_i = 0$.

Point i	1	2	3	4	5	6	7	8	9
α_i, Laplace Eq.	1/20	1/5	1/20	1/5	1/5	1/20	1/5	1/20	**−1**
α_i, Neumann B.C.	11/30Δh	**-46/30Δ*h***	11/30Δh	9/30Δh	9/30Δh	1/30Δh	4/30Δh	1/30Δh	0

3.1 Irregular cell

The irregular cell is illustrated in Fig. 1. It is constructed by the neighboring points of the center point C, where the Laplace equation is to be discretized. The center point is also called 'cell center'. Any point, close to the center point, can be regarded as a neighboring point. Assuming that the irregular cell with the center point C consists of m neighboring points P_i $(i = 1, 2, \dots, m)$, we set up a local Cartesian coordinate system to discretize the Laplace equation

$$\bar{\mathbf{x}}_i = (\mathbf{x}_i - \mathbf{x}_C)/L, \tag{16}$$

where L is the characteristic size of the irregular cell and can be set to be $L = \sum_{i=1}^{m} |\mathbf{x}_i - \mathbf{x}_C| / m$. For simplicity, the bar symbol in Eq. (16) will be dropped in the following presentation. Near the cell center, the harmonic function can be approximated by the truncated Fourier series of order k

$$\varphi(r, \theta) = A_0 + \sum_{n=1}^{k} [A_n r^n \cos n\theta - B_n r^n \sin n\theta]. \tag{17}$$

Eq. (17) can be rewritten as

$$\varphi(x, y) = \sum_{j=1}^{2k+1} b_j h_j(x, y). \tag{18}$$

The value of the approximated harmonic function at the neighboring points are forced to be the corresponding point value φ_i $(i = 1, 2, \dots, m)$, which results in the following matrix equation

$$H\underline{b} = \underline{\varphi}. \tag{19}$$

Here, H is the matrix with elements $H_{i,j} = h_j(x_i, y_i)$, $\underline{b} = [b_1, b_2, \dots, b_{2k+1}]^T$ and $\underline{\varphi} = [\varphi_1, \varphi_2, \dots, \varphi_m]^T$. To guarantee that Eq. (19) is solvable, m should be sufficiently larger than $2k+1$. Then Eq. (19) is solved by the least-square method, i.e.,

$$\underline{b} = [H^T H]^{-1} H^T \underline{\varphi}. \tag{20}$$

At the center point, $\varphi(x, y)$ should be equal to the point value φ_C, resulting in $\varphi_C = b_1$. Then, through Eq. (20), φ_C is related to the neighboring point values

$$-\varphi_C + \sum_{i=1}^{m} \alpha_i \varphi_i = 0, \tag{21}$$

which represents the discretized Laplace equation at the center point. Here, $[\alpha_1, \alpha_2, \dots, \alpha_m]$ is the first row of the matrix $[H^T H]^{-1} H^T$. The harmonic polynomial method based on irregular cells, regarded as the generalization of HPC, has great freedom to choose the order of the harmonic polynomial basis and to select the neighboring points.

We consider two Dirichlet problems, i.e., Case I and Case II shown in Fig. 3. The value of φ along the boundaries are specified by the harmonic function $\varphi(x, y) = \frac{\cosh[k(y+h)]}{\cosh k} \sin kx$, where $h = 1$ and $k = 2\pi$. The computational domains can be meshed based on the structured gird, whose points (nodes) are addressed by a two-dimensional matrix (i, j). The resulting elements of Case I are perfect squares. In contrast, the elements of Case II are distorted. To discretize the Laplace equation at the point (I, J), HPC adopts the nine points of $\mathrm{I}-1 \le \mathrm{i} \le \mathrm{I}+1$ and $\mathrm{J}-1 \le \mathrm{j} \le \mathrm{J}+1$, which are associated with the four neighboring elements. The corresponding numerical results are presented in Table 2, where the numerical error is measured by the maximum norm, i.e., $error = \max_{i,j} \left| \varphi_{i,j}^{(N)} - \varphi_{i,j}^{(E)} \right|$. Here, $\varphi_{i,j}^{(N)}$ denotes the numerical solution at point (i, j) and $\varphi_{i,j}^{(E)}$ the exact solution. When the numerical error converges at the rate of $(\Delta h)^{\beta}$, we say that the convergence order of the numerical results is β. The theoretical convergence order of HPC is 4 to 5, because it adopt incomplete harmonic polynomials of order 4. HPC works very well on the grid with square elements, resulting in the highly accurate results and even the super convergence order (larger than 5). However, when the grid elements are distorted as shown in Case II, the results become poor and even can not be improved by reducing the grid size. This situation can be resolved by the proposed method based on irregular cells. Keeping in mind that the number of the neighboring points, m, should be sufficiently larger than $2k+1$ (k is the order of the harmonic polynomials), we can construct the 4th-order irregular cells by the points (i, j) of $\max(1, \mathrm{I}-2) \le \mathrm{i} \le \min(N+1, \mathrm{I}+2)$ and $\max(1, \mathrm{J}-2) \le \mathrm{j} \le \min(N+1, \mathrm{J}+2)$. Table 2 shows: the present method works well on both the square elements and the distorted elements; the corresponding results are highly accurate and achieve an averaged converge order of about 4.5 (which could be accurate enough).

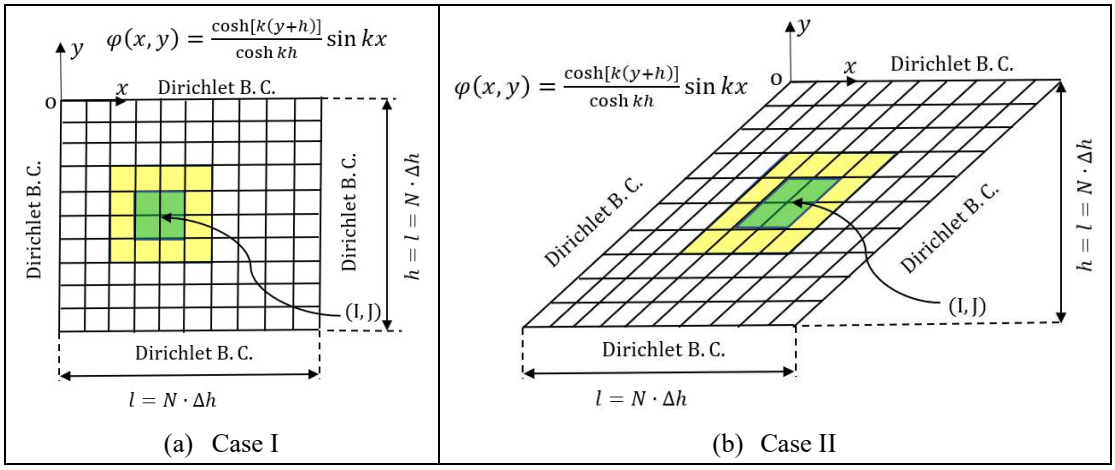
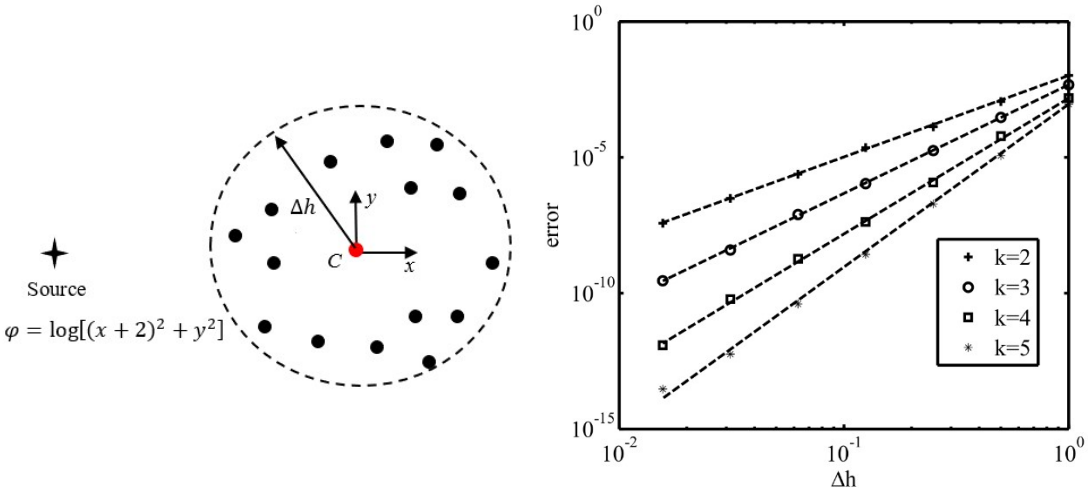


Fig. 3. Definition of two problems with Dirichlet boundary conditions. The computational domain is meshed by the structured grid, whose nodes are addressed by a two-dimensional matrix (i, j). The grid points on the left, right, bottom and top boundaries correspond to i=1, i=*N*+1, j=1 and j=*N*+1 respectively.

Table 2. Convergence study of the numerical solution of the Dirichlet problems defined by Fig. 3.The numerical error is measured by the maximum norm. 'Order' denotes the convergence order of the numerical results.

Grid		Case I				Case II			
$(l=h=N\cdot\Delta h)$		HPC		Irregular cell		HPC		Irregular cell	
Δh	N	Error	Order	Error	Order	Error	Order	Error	Order
0.04	25	1.53e-8	–	3.04e-5	–	1.03e-0	–	2.82e-4	–
0.02	50	2.39e-10	6.0	1.21e-6	4.65	1.03e-0	0	1.12e-5	4.65
0.01	100	3.71e-12	6.0	4.24e-8	4.83	1.03e-0	0	3.90e-7	4.84
0.005	200	7.00e-14	5.73	1.41e-9	4.91	1.03e-0	0	1.83e-8	4.41
0.0025	400	5.99e-13	–	4.53e-11	4.96	1.03e-0	0	1.24e-9	3.88

The harmonic polynomial method based on irregular cells can also be implemented on unstructured grids, for instance the grid system presented in Section 4. Even it may work for the randomly-distributed neighboring points. We consider the potential flow induced by a two-dimensional source. The irregular cell centered at the origin is built up by the randomly-distributed neighboring points, whose polar coordinates (r_i, θ_i) are two random series $\{r_i\} \subset [\Delta h/2,\ \Delta h]$ and $\{\theta_i\} \subset [0,\ 2\pi]$. The point values φ_i $(i = 1, 2, \dots, m)$ are specified by a two-dimensional source located at $(-2, 0)$, i.e., $\log[(x_i + 2)^2 + y_i^2]$. From Eq. (21), the numerical value at the center point is expressed as $\varphi_C^{(N)} = \sum_{i=1}^{m} \alpha_i \varphi_i$, which is compared to the exact value $\varphi_C^{(E)} = 2\log 2$ by measuring the error $\left|\varphi_C^{(E)} - \varphi_C^{(N)}\right|$. Theoretically, the error converges at the rate of $(\Delta h)^{k+1}$, which is consistent with the truncated error of the approximation by the harmonic polynomials of order k. This is confirmed by the results shown in Fig. 4, where 15 randomly-distributed neighboring points are used to construct the irregular cell and the numerical value $\varphi_C^{(N)}$ is the average value of 40 random cells.



(a) sketch (b) results

Fig. 4. Convergence analysis for the numerical interpolation by the harmonic polynomial method based on the irregular cells with randomly-distributed neighboring points. (a) sketch of interpolating φ_C by φ_i at the randomly-distributed neighboring points (the black points). (b) numerical results: Δh is the grid size; the numerical error is computed as $\left|\varphi_C^{(E)} - \varphi_C^{(N)}\right|$; k is the order of harmonic polynomials; the dashed lines represent the theoretical trend of $(\Delta h)^{k+1}$.

3.2 Normal derivative on boundaries

The representation of the normal derivative of functions on boundaries is directly related to the Neumann boundary conditions on the body boundaries and the normal velocity of potential flows on the free-surface boundaries. It plays a key role in the accurate and stable modeling of the highly-nonlinear free-surface potential flow with and without moving bodies, which is often numerically challenging. For example, numerical instabilities may arise on the free-surface boundaries when nonlinear free-surface flows are solved in time domain [27]. For the accurate representation of free-surface/body boundaries, the boundary-fitted grid system can be an option. However, the generation of high-quality boundary-fitted grids is commonly a cumbersome and time-consuming task especially for the highly-deformed free surface or the body with complex geometry. An alternative solution is to adopt immersed boundary methods [28], which allows the bodies to move freely over a fixed (Cartesian) background grid [24]. However, new numerical challenges, e.g. spurious force oscillations [25, 29], may arise. To overcome these numerical difficulties, the free-surface deformation and the body motion should be tracked in a proper way, which requires the accurate evaluation of the normal derivatives of potential functions. In the present study, we will propose an accurate numerical scheme for the normal derivative of harmonic functions. It is able to evaluate the normal derivative on complex boundaries in a convenient way. Together with the grid system and the free-surface tracking method presented in the following sections, the proposed method can solve highly-nonlinear free-surface potential flows with and without moving bodies in an accurate and stable manner.

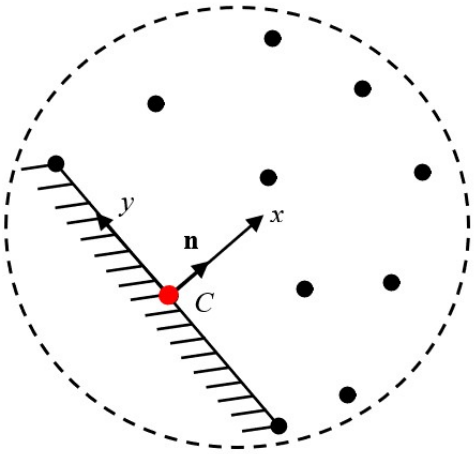


Fig. 5. Sketch of the local coordinate system and neighboring points for evaluating the normal derivative of harmonic functions at the point *C* on the boundary. The black points denote the neighboring points of *C*.

To evaluate the normal derivative of harmonic functions at the point *C* on the boundary, a local coordinate system with the *x*-axis coinciding with the normal vector is introduced, which is illustrated by Fig. 5. The neighboring field points indexed by $i = 1, 2, \dots, m$, are chosen to construct an irregular cell similar to that we did before. Locally, the harmonic function can be approximated by the harmonic polynomials up to order *k*

$$\varphi(x, y) = \varphi_C + \sum_{j=2}^{2k+1} b_j h_j(x, y). \tag{22}$$

By forcing the above equation equal to the point value φ_i $(i = 1, 2, \dots, m)$, we have

$$\tilde{H}\underline{\tilde{b}} = \underline{\tilde{\varphi}}, \tag{23}$$

where $\tilde{H}$ is the matrix with elements $\tilde{H}_{i,j} = h_{j+1}(x_i, y_i)$, $\underline{\tilde{b}} = [b_2, b_3, \dots, b_{2k+1}]^T$ and $\underline{\tilde{\varphi}} = [\varphi_1 - \varphi_C, \varphi_2 - \varphi_C, \dots, \varphi_m - \varphi_C]^T$. Solving Eq. (23) by the least-square method results in

$$\underline{\tilde{b}} = \left[\tilde{H}^T \tilde{H}\right]^{-1} \tilde{H}^T \underline{\tilde{\varphi}}. \tag{24}$$

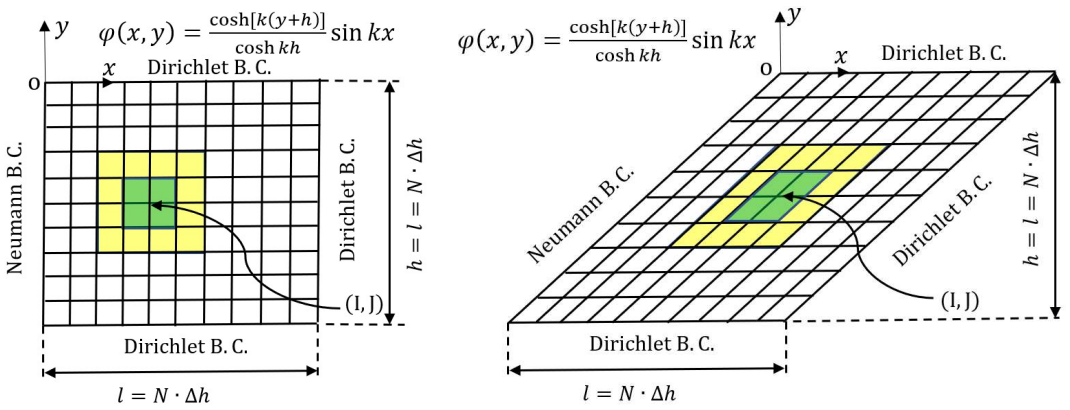
Immediately, we can express the normal derivative at point *C* as $\partial\varphi/\partial n = b_2$ or

$$\left(\frac{\partial \varphi}{\partial n}\right)_C = -(\sum_{i=1}^{m} \tilde{\alpha}_i)\varphi_C + \sum_{i=1}^{m} \tilde{\alpha}_i \varphi_i, \tag{25}$$

where $[\tilde{\alpha}_1, \tilde{\alpha}_2, \dots, \tilde{\alpha}_m]$ is the first row of the matrix $\left[\tilde{H}^T \tilde{H}\right]^{-1} \tilde{H}^T$. If the dimensionless coordinates expressed by Eq. (16) are used, the coefficients $\tilde{\alpha}_i$ should be divided by the reference length *L*.

We consider two problems with a mixed boundary condition, i.e., Case I and Case II shown in Fig. 6. Compared to the two Dirichlet problems shown in Fig. 3, the only difference is that the left boundary of the present problems is the Neumann boundary condition. Table 3 presents the numerical results by HPC and the present methods based on the irregular cells, which are constructed by the points (i, j) of $\max(1, \mathrm{I} - 2) \le \mathrm{i} \le \min(N + 1, \mathrm{I} + 2)$ and $\max(1, \mathrm{J} - 2) \le \mathrm{j} \le \min(N + 1, \mathrm{J} + 2)$ for discretizing the Laplace equation and the Neumann boundary condition at a given point (I, J). On the grid system with square elements, HPC works well and achieves the convergence order of 4. The accuracy of the numerical solution is lower than that of the Dirichlet problem, which implies that the Neumann B.C. dominates the numerical error. Similar to the Dirichlet problem, HPC can not work properly on the distorted elements. In contrast, the present method works well on both the square elements and the

distorted element and achieves the convergence order of 4. We note that, on the grid system with square elements, the accuracy of the present method is slightly less than that of HPC.



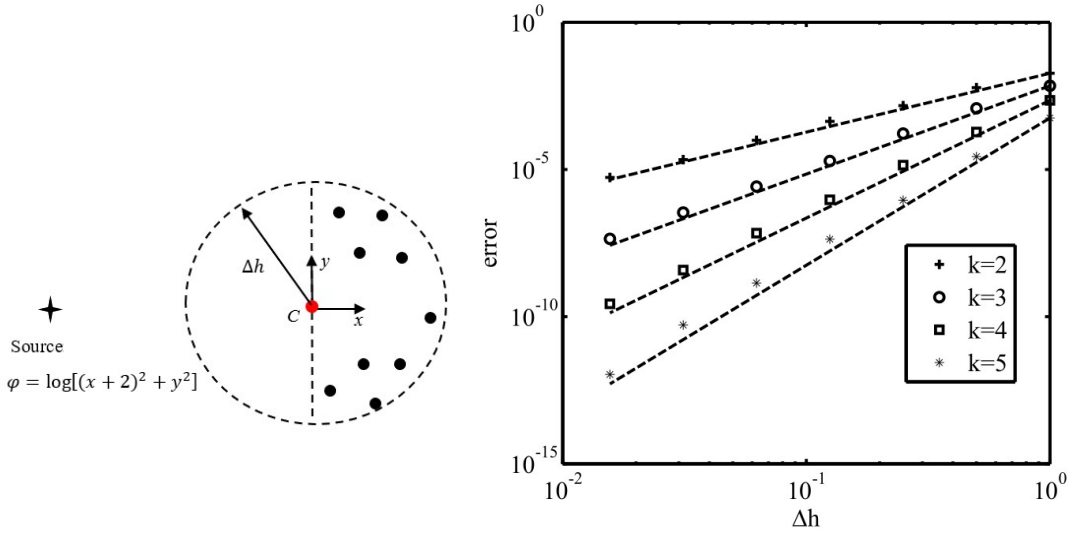
(a) Case I (b) Case II

Fig. 6. Definition of two problems with a mixed boundary condition. The computational domain is meshed by the structured grid, whose nodes are addressed by a two-dimensional matrix (i, j). The grid points on the left, right, bottom and top boundaries correspond to i=1, i=*N*+1, j=1 and j=*N*+1 respectively.

Table 3. Convergence study of the numerical solution of the mixed B.C. problems defined by Fig. 6. The numerical error is measured by the maximum norm. 'Order' denotes the convergence order of the numerical results.

Grid		Case I				Case II			
($l = h = N \cdot \Delta h$)		HPC		Irregular cell		HPC		Irregular cell	
Δh	N	Error	Order	Error	Order	Error	Order	Error	Order
0.04	25	8,83e-6	–	1.43e-4	–	2.57e-0	–	3.66e-4	–
0.02	50	5.56e-7	4.0	8.87e-6	4.0	2.74e-0	0	2.32e-5	4.0
0.01	100	3.48e-8	4.0	5.40e-7	4.0	2.80e-0	0	1.48e-6	4.0
0.005	200	2.17e-9	4.0	3.31e-8	4.0	2.84e-0	0	9.32e-8	4.0
0.0025	400	1.36e-10	4.0	2.04e-9	4.0	2.84e-0	0	5.86e-9	4.0

Similar to the previous study, the accuracy of the proposed method is investigated for the irregular cell with randomly-distributed neighboring points. We still consider the potential flow induced by a two-dimensional source. The irregular cell centered at the origin is built up by the randomly-distributed neighboring points, whose polar coordinates (r_i, θ_i) are two random series $\{r_i\} \subset [\Delta h/2,\ \Delta h]$ and $\{\theta_i\} \subset [-\pi/2,\ \pi/2]$. The point values φ_i $(i = 1, 2, \dots, m)$ are specified by a two-dimensional source located at $(-2, 0)$, i.e., $\log[(x_i + 2)^2 + y_i^2]$. Through Eq. (25), we evaluate the numerical value $[\partial\varphi/\partial x]_C^{(N)}$, which is compared to the exact value $[\partial\varphi/\partial x]_C^{(E)} = 1$ by measuring the error $\left|[\partial\varphi/\partial x]_C^{(E)} - [\partial\varphi/\partial x]_C^{(N)}\right|$. Here, 15 randomly-distributed neighboring points are used to construct the irregular cell and $[\partial\varphi/\partial x]_C^{(N)}$ is the average value of 40 random cells. The numerical errors, shown in Fig. 7, converge at the rate of $(\Delta h)^k$ approximately, where k is the order of the harmonic polynomials indicated in Eq. (22).



(a) sketch (b) results

Fig. 7. Convergence analysis for the numerical derivative by the harmonic polynomial method based on the irregular cells with randomly-distributed neighboring points. (a) sketch of evaluating the derivative $\partial\varphi/\partial x$ at the center point C with randomly-distributed neighboring points (the black points). (b) numerical results: Δh is the grid size; the numerical error is computed as $\left|[\partial\varphi/\partial x]_C^{(E)} - [\partial\varphi/\partial x]_C^{(N)}\right|$; k is the order of the harmonic polynomials; the dashed lines represent the reference trend of $(\Delta h)^k$.

4. Grid system

For harmonic polynomial methods, a grid system is required to discretize the computational domain. A boundary-fitted grid system and a Cartesian grid system are two options. The boundary-fitted grid system can well represent the boundary but commonly results in distorted elements. Furthermore, with the presence of complex boundaries, the generation of high-quality boundary-fitted grids can be a cumbersome and time-consuming task. The Cartesian grid system can result in perfect elements (i.e., square elements) but needs to deal with the embedded boundary in a proper way. As shown in the previous section, the HPC works very well on square elements but may become inaccurate on distorted elements, which are often encountered by complex boundaries. The proposed harmonic polynomial methods based on irregular cells work well on both square elements and distorted elements. Both the HPC on distorted elements and the irregular cell require extra computational efforts to build up the discretized Laplace equation. Based on these, we decide to adopt the Cartesian grid system with square elements. The computational domain is embedded in the grid system. The boundary of the computational domain is represented by a number of marker points. The value of harmonic functions is stored at the center of the cells and the boundary conditions are satisfied on the marker points. Inside the computational domain and away from the boundary, the standard HPC, which is highly accurate and does not require extra computational effort to build up the local matrix system, is used. The irregular cell, which is also accurate, is only used to the field points close to the boundary, where it is difficult to construct the high-quality HPC. Although it requires extra computational effort to build up the local matrix for the irregular cell, the operation is only limited to the field points close to the boundary, which is a small group compared to the whole field points. Fig. 8 illustrates applying the proposed grid system to solve the mixed B.C. problem defined by Fig. 6-(b). The standard HPC can be constructed for the center points of the green cells. The irregular cells are used for the center points of the yellow cells and the markers on the Neumann boundary. Fig. 9 presents the numerical results, where the 4th-order

harmonic polynomials are used for the irregular cells. With decreasing the grid size Δh, the numerical error converges at the rate of $(\Delta h)^4$. Compared to the boundary-fitted grid (as shown in Fig. 6-(b)), the Cartesian grid with the proposed harmonic polynomial method significantly improves the accuracy and reduces the computational time. The ratio of the computational time of building up the matrix system for the Cartesian grid to that for the boundary-fitted grid decreases dramatically with reducing the grid size. It is because the ratio of the field points, adopting the irregular cell, to the whole field points becomes smaller and smaller with reducing the grid size.

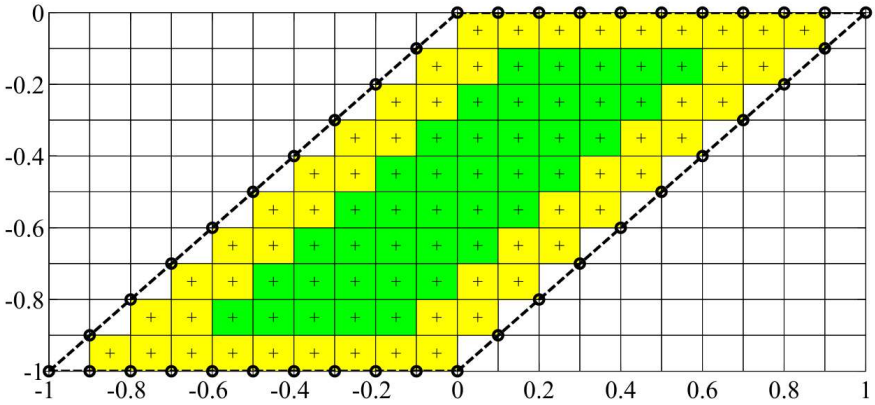


Fig. 8. Sketch of the Cartesian grid for solving the mixed B.C. problem defined by Fig. 6-(b). The boundary is represented by the markers of 'circle'. The center of active cells is marked by the 'cross' symbol.

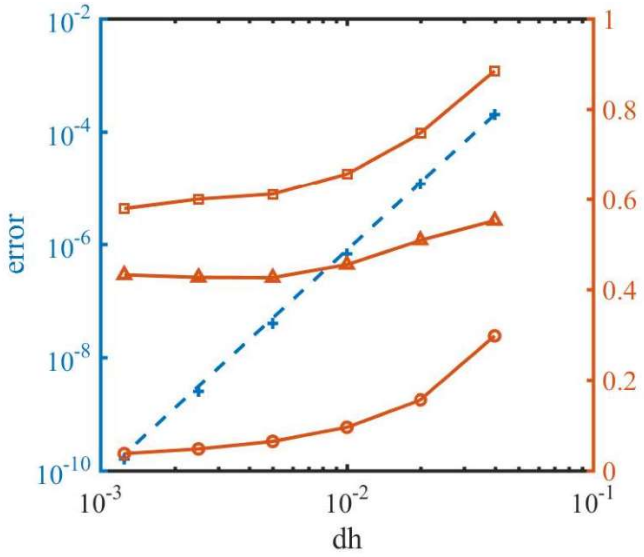


Fig. 9. Convergence analysis for the numerical solution of the mixed B.C. problem defined by Fig. 6-(b) through the present method based on the Cartesian grid. The blue 'cross' denotes the numerical error measured by the maximum norm. The dashed blue line represents the reference trend of $(\Delta h)^4$. The brown symbols represent the ratio of the numerical result based on the Cartesian grid to that based on the boundary-fitted grid shown in Fig. 6-(b): the 'triangle' denotes the numerical error; the 'circle' denotes the computational time of building up the matrix system; the 'square' denotes the computational time of solving the matrix system by the direct sparse matrix solver, PARDISO (referring to https://www.pardiso-project.org).

In the region where the flow changes strongly or on the highly-curved boundaries, finer grids are required to properly represent the flow. The uniform refinement of the grid system will result in the dramatic growth of the computational cost, because the number of the field points is approximately proportional to $(\Delta h)^{-2}$ in the two-dimensional space and $(\Delta h)^{-3}$ in the three-dimensional space. A rational solution could be adopting the method of local refinement, i.e., the grid system is refined only in the regions of interest. A natural refinement procedure is successively subdividing the cells of interest into four equally sized quadrants, which can be represented by quadtrees. In the following, we will detail the data structure of the grid system, the refinement procedure and the basic operations such as neighbor searches.

4.1 Data structure of the grid system

In a quadtree, a cell may be the *parent* of four *children*. The *root cell* is the base of a quadtree and a *leaf* is a cell without any child. The *depth* (*level*) of a cell is defined by starting from 1 for the root cell and by adding one every time a group of descendant children is appended. The original Cartesian grid consists of the root cells, which are addressed by a two-dimensional matrix (i,j) with $1 \le \mathrm{i} \le N_x^{(root)}$ and $1 \le \mathrm{j} \le N_y^{(root)}$. The locally-refined Cartesian grid system can be regarded as the two-dimensional matrix of quadtrees. We introduce the concept of *neighboring cell*. The neighboring cells of a cell A are the contacting cells (including Cell A itself), whose depth are less than or equal to the depth of Cell A. The neighboring cells of Cell A can be addressed by $(-1{:}\,1, -1{:}\,1)$, where $(0{,}0)$ corresponds to Cell A itself. For leaves, the concept of *neighboring leaf* is also introduced. The neighboring leaves of a leaf are all the contacting leaves (excluding itself), which can be represented by an integer array. In a balanced grid system, where the depths of all neighboring leaves do not differ by more than one, a leaf has maximum 12 neighboring leaves. The balanced grid system tends to avoid the strong variance of the grid size, which is adopted in the present study. To explain these concepts, we consider an example shown in Fig. 10. The root cell 4 has four children: $cell(4).child(-1,-1) = 10$, $cell(4).child(-1{,}1) = 12$, $cell(4).child(1,-1) = 11$ and $cell(4).child(1{,}1) = 13$. It is the parent of Cells 10, 11, 12 and 13, e.g. $cell(10).parent = 4$. Cell 4 has a depth of 1, i.e., $cell(4).depth = 1$, and $cell(10).depth = 2$. The neighboring cells of Cell 5 are:

$cell(5).neighboringCell(-1,-1) = 1,$
$cell(5).neighboringCell(-1,0) = 4,$
$cell(5).neighboringCell(-1,1) = 7,$
$cell(5).neighboringCell(0,-1) = 2,$
$cell(5).neighboringCell(0,0) = 5,$
$cell(5).neighboringCell(0,1) = 8,$
$cell(5).neighboringCell(1,-1) = 3,$
$cell(5).neighboringCell(1,0) = 6,$
$cell(5).neighboringCell(1,1) = 9.$

While, the neighboring leaves of Cell 5 are

$cell(5).neighboringLeaf = [1{,}2{,}3{,}6{,}7{,}8{,}9{,}11{,}13]$with $cell(5).N_{neighboringLeaf} = 9.$

The neighboring cells of Cell 13 are:

$cell(13).neighboringCell(-1,-1) = 10,$
$cell(13).neighboringCell(-1,0) = 12,$
$cell(13).neighboringCell(-1,1) = 7,$
$cell(13).neighboringCell(0,-1) = 11,$

$cell(13).neighboringCell(0,0) = 13,$
$cell(13).neighboringCell(0,1) = 7,$
$cell(13).neighboringCell(1,-1) = 5,$
$cell(13).neighboringCell(1,0) = 5,$
$cell(13).neighboringCell(1,1) = 8.$

While, the neighboring leaves of Cell 13 are

$cell(13).neighboringLeaf = [5,7,8,10,11,12]$ with $cell(13).\ N_{neighboringLeaf} = 6$.

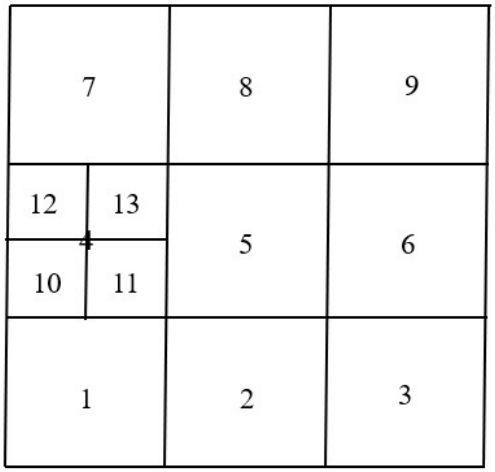


Fig. 10. Example of a local-refined Cartesian grid.

For the cells on the boundary of the Cartesian grid, the neighboring cells can be represented as, for example,

$cell(1).neighboringCell(-1,-1) = 0,$
$cell(1).neighboringCell(-1,0) = 0,$
$cell(1).neighboringCell(-1,1) = 0,$
$cell(1).neighboringCell(0,-1) = 0,$
$cell(1).neighboringCell(0,0) = 1,$
$cell(1).neighboringCell(0,1) = 4,$
$cell(1).neighboringCell(1,-1) = 0,$
$cell(1).neighboringCell(1,0) = 2,$
$cell(1).neighboringCell(1,1) = 5.$

Here, the index of the neighboring cell equal to zero means that the corresponding cell does not exist. Further, the *distance* between Leaf A and Leaf B is defined as the minimum number of crossing the leaf cell boundary from A to B. For example, the distance between Leaf 13 and Leaf 8 is one and the distance between Leaf 13 and Leaf 9 is two.

The boundary of the computational domain is represented by marker points (or simply, markers). The basic properties of markers are the coordinates and the associated normal vector of the boundary, which can be represented as $marker(*).x$, $marker(*).y$, $marker(*).n_x$ and $marker(*).n_y$. We also introduce to the markers the property of *depth*, $marker(*).depth$, which matches with the size of the associated boundary elements, i.e., $(\sqrt{2}\Delta h)2^{1-depth} \sim l$. Here, Δh is the grid size of root cells and l is the size of the boundary elements. Then, the depth of the markers can be approximated as

$$marker(*).depth = \text{int}\left[\frac{\ln(\sqrt{2}\Delta h/l)}{\ln(2)}\right] + 1, \tag{26}$$

where the function $\mathrm{int(x)}$ rounds x to the nearest integer less than or equal to x. This property will be used for the grid refinement procedure: the adjacent leaf cells are to be refined until the depth of the new leaf cells is equal to the depth of the marker. It is based on the principle that the grid size should match with the size of the boundary element. We can distribute some 'virtual' markers with a prescribed depth in the region of interest for the grid refinement. Another important property of the marker is the location, $marker(*).location$, indicating the leaf cell where the marker locates.

4.2 Grid refinement procedure

The basic principle of the grid refinement procedure is successively subdividing the cells into four equally sized quadrants until the depth of all markers matches with the depth of the leaf cells where the markers locate. In the grid refinement procedure, a level-by-level strategy is adopted: only after the partition of the targeted cells to four children cells is completed on level n, the partition on level $n+1$ can start. On the level- n partition, the cells, with the depth $\leq n$ and close to the markers of $marker(*).depth > n$, are targeted for the partition. There are $maxDepth - 1$ levels of partition, where $maxDepth$ denotes the maximum depth of all markers. The detailed grid-refinement procedure is as follows:

(1) The cell index is starting from the root cells and the root cell $(\mathrm{i, j})$ can be indexed as, for example, $(\mathrm{i}-1) * N_x^{(root)} + j$. Specify the property of $cell(*).neighboringCell(-1:1, -1:1)$ for all root cells. Locate all markers, i.e., specify the property of $marker(*).location$.

(2) Partition all targeted cells to four children cells on level n.

a) Target the cells to be partitioned. The cells close to the markers of $marker(*).depth > n$ are the candidates to be partitioned. For example, Assume the depth of marker A is larger than n. The location of marker A is obtained by $p = marker(A).location$. Any cell q, with $cell(q).depth \leq n$ and $distance(cell(p), cell(q)) \leq N$, is targeted for the partition. Here, $distance\big(cell(p), cell(p)\big) = 0$, $distance\big(cell(p), cell(q)\big) = 1$ if $p \neq q$ and $q \in cell(p).neighboringCell$, and $distance\big(cell(p), cell(q)\big) = m + 1$ if $q \notin cell(p).neighboringCell$ (m is the minimum number of the intermediate cells $\{p_1, p_2, \dots, p_r\}$ connecting p to q by $p_1 \in cell(p).neighboringCell$, $p_2 \in cell(p_1).neighboringCell$, …, $q \in cell(p_r).neighboringCell$). Commonly, $N = 2 \sim 3$ is adopted.

b) Partition the target cells if they have not been partitioned yet by checking whether $cell(*).child(-1, -1)$ is equal to 0.

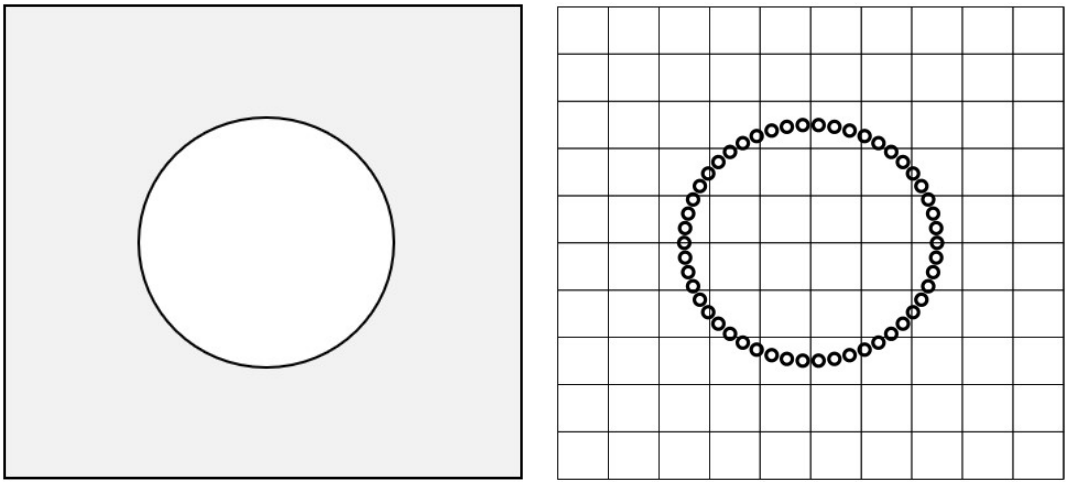
(3) Update the location of the markers. The traditional point-location method, which recursively compares the position of the point to that of cell centers, is used to determine the leaf cell that contains a given marker *A*. First, set $p = marker(A).location$. If $cell(p).child(-1, -1) \neq 0$, set $p = cell(p).child(sign(x - x_c),\ sign(y - y_c))$, where (x, y) is the coordinates of Marker *A* and (x_c, y_c) is the center of $cell(p)$. Repeat the procedure until the $cell(p)$ becomes a leaf. Finally, set $marker(A).location = p$.

(4) Specify $cell(*).neighboringCell$ for the new cells generated on the level- n partition, which can be accomplished by the algorithm presented in Section 4.3.

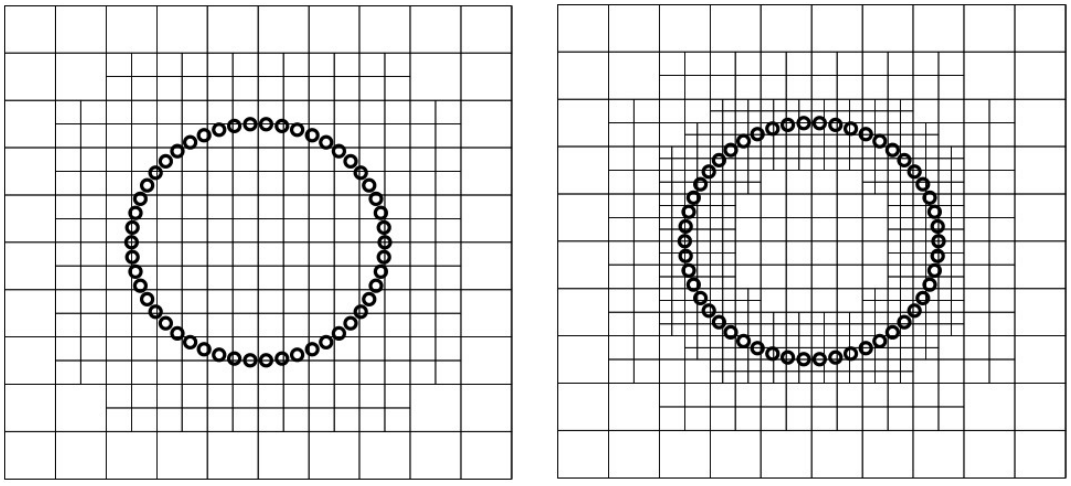
(5) Go to step (2) until the $(maxDepth - 1)$-level partition is finished.

Fig. 11 illustrates the grid refinement procedure for the domain of the 2×2 square, of which a unit circle is hollowed out. As shown in Fig. 11-(a), the square is represented by the Cartesian grid with

10×10 cells and the boundary of the unit circle is represented by 50 equally-spaced markers. According to Eq. (26), the depth of the markers is set to be 3. In the original Cartesian grid, the cells, close to the markers, should be successively partitioned until the depth of the refined cells matches that of the markers. Fig. 11-(b) shows the refinement procedure by using $N = 1$ (where N is defined in the step of (2)-(a)) and Fig. 11-(c) the refinement procedure by $N = 2$. In general, higher N results in more gradual refinement (therefore the grid of better quality) but larger numbers of cells. Commonly, $N = 2 \sim 3$ is used.



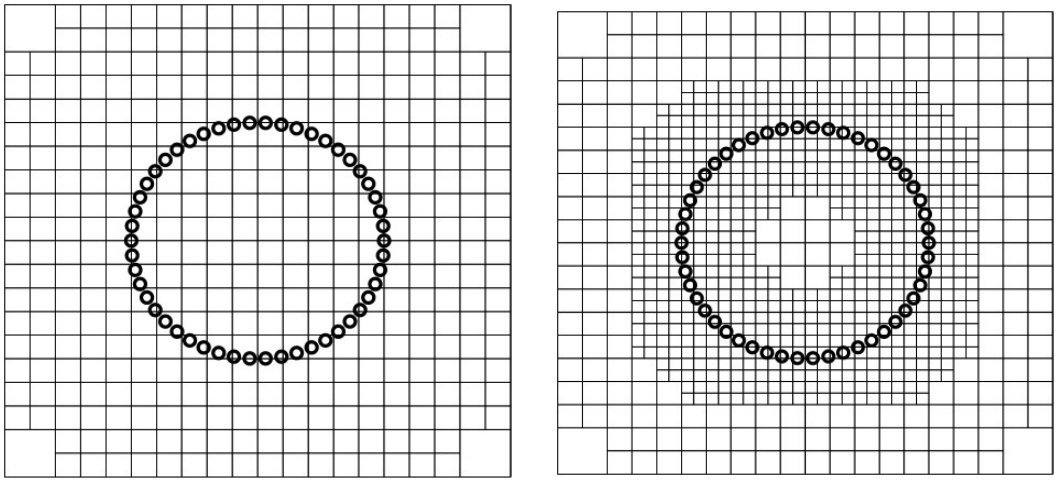
(a) Computational domain and the original Cartesian grid system.



after the level-1 partition

after the level-2 partition

(b) $N = 1$



after the level-1 partition　　after the level-2 partition

(c) $N = 2$

Fig. 11. Illustration of the grid refinement procedure for the square domain with a hollowed circle. The square domain of 2×2 is represented by the Cartesian grid with 10×10 elements; the boundary of the unit circle is represented by 50 equally-spaced markers.

4.3 Neighbor searches

Neighbor searches are the basic operations, which are used in the generation of the locally-refined grid and also in the discretization of the governing equations. First, we present the algorithm of searching the neighboring cells, i.e., specifying $cell(*).neighboringCell$ in the grid-refinement procedure. Based on the preset condition that $cell(*).neighboringCell$ has been specified for all cells with the depth of n, we try to find the neighboring cells for the cells with the depth of $n + 1$. Without loss of generality, Fig. 12 presents the sketch of searching neighboring cells of Cell C.

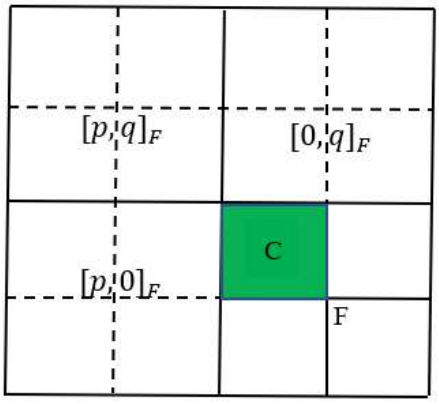


Fig. 12. Sketch of searching the neighboring cells of Cell C. F denotes the parent of Cell C.

We adopt the round brackets denoting the children cells, i.e., $cell(*).child(p,q) = (p,q)_*$, and the square brackets denoting the neighboring cells, i.e., $cell(*).neighboringCell(p,q) = [p,q]_*$. The neighboring cells of Cell C can be found by the following steps:

(1) Compute the index of Cell C in the children cells of its parent cell F by

$$(p,q)_F = (sign(x_C - x_F), sign(y_C - y_F))_F.$$

Immediately, we have the four neighboring cells of Cell C

$$[i,j]_C = (p + 2 * i, q + 2 * j)_F \text{ with } i = 0 \text{ or } -p \text{ and } j = 0 \text{ or } -q.$$

(2) The cell $r = [p, q]_F$ (or one of its children) is a neighboring cell of Cell C
$[p, q]_C = r$ (if r is a leaf cell) or $[p, q]_C = (-p, -q)_r$.

(3) The cell $r = [p, 0]_F$ (or two of its children) corresponds to two neighboring cells of Cell C
$[p, j]_C = r$ (if r is a leaf cell) or $[p, j]_C = (-p, q + 2 * j)_r$, where $j = 0\ or - q$.
Similarly, we can find two other neighboring cells related to $[0, q]_F$.

The balanced grid system, where the depths of all neighboring leaves do not differ by more than one, is used in the present study. Once the neighboring cells become known, it is easy to search the neighboring leaves. We consider any neighboring cell of Leaf C, i.e., $r = [p, q]_C$. If r is a leaf cell, it is a neighboring leaf of C. Otherwise, its child, $(f(p), f(q))_r$, is the neighboring leaf of C. Here, $f(p)$ is the multivalued function

$$f(p) = \begin{cases} \pm 1, & for\ p = 0 \\ -p, & for\ p = \pm 1 \end{cases}.$$

4.4 Distributing field points

The field points, where the harmonic function is solved, are distributed at the center of the leaf cells inside the computational domain and at the markers on the boundary. The leaf cell, of which the center is attached by a field point, is defined as an 'active' cell. To identify active cells, the leaf cells are divided into three groups

(i) the interfacial group: the leaf cells close to the markers on the boundary, i.e., the leaf cells, *, with $distance\big(cell(*), cell(A)\big) \leq M$, where A is the location of a marker and M is a prescribed parameter (commonly 1).

(ii) the inner group: the leaf cells which are inside the computational domain and exclude the cells in the interfacial group.

(iii) the outer group: the leaf cells which are outside the computational domain and exclude the cells in the interfacial group.

The inner group and the outer group are separated from each other by the interfacial group. Obviously, any leaf cell in the inner group can be identified as an active cell and the leaf cells in the outer group are "inactive". A leaf cell in the interfacial group can be active or inactive. Consider a marker locating at the leaf cell A, illustrated in Fig. 13. The dashed line, orthogonal to the normal vector of the boundary, is regarded as the local approximation of the curved boundary. Then, the signed distance of the cell center (x_C, y_C) to the boundary is calculated by

$$d = n_x \cdot (x_C - x_0) + n_y \cdot (y_C - y_0), \tag{27}$$

where (n_x, n_y) denotes the interior normal vector and (x_0, y_0) the coordinates of the marker. $d < 0$ means that the cell is outside the computational domain and therefore becomes inactive. In the computer program, we adopt the condition of $d < \varepsilon$ (ε is a small positive value) instead of $d < 0$. Special attention is paid to the cell A, where the marker locates. We can set Cell A to be a potential active cell if the center of Cell A is far enough away from the marker, e.g. the distance l is larger than a quarter of the cell size (see Fig. 13-(a)). Even we directly set Cell A to be inactive (see Fig. 13-(b)). Based on our experiences, the latter improves the stability of the numerical solver in the time-domain simulations.

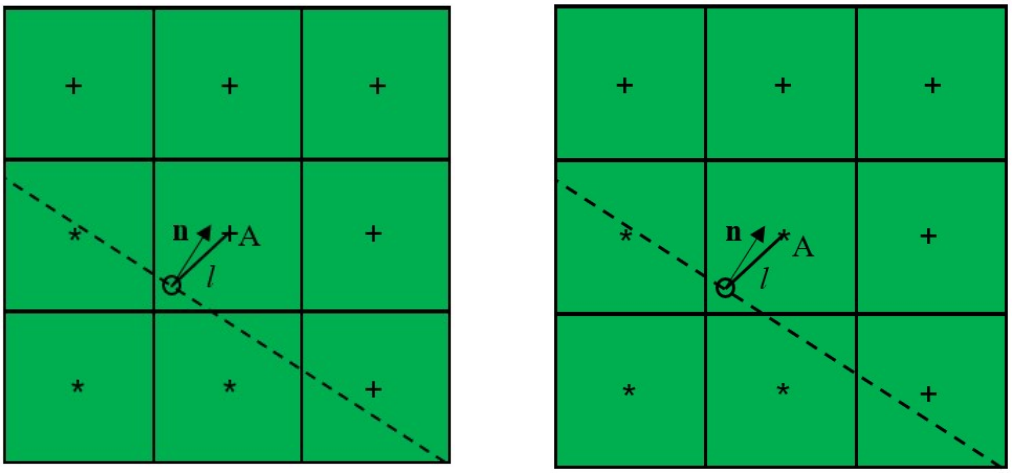


Fig. 13. Identifying inactive cells. The "circle" denotes the marker point, which locates at Cell A. **n** denotes the interior normal of the boundary. The dashed line represents the local approximation of the boundary. The green color indicates that these cells are in the interfacial group. "*" coincides with the center of the inactive cells. "+" coincides with the center of the possible active cells. In (a), Cell A is set to be a potential active cell if l is large enough (e.g., larger than a quarter of the cell size); in (b), Cell A is set to be inactive.

Fig. 14 illustrates the distribution of the field points for the domain of the 2×2 square, of which a unit circle is hollowed out. The square is represented by the Cartesian grid with 10×10 cells and the boundary of the unit circle is represented by 25 equally-spaced markers. According to Eq. (26), the depth of the markers is set to be 2. After the grid refinement procedure, the leaf cells are classified into three groups: the interfacial group, close to the markers on the boundary, is indicated by the green color; the inner group is indicated by the blue color; the outer group is indicated by the white color. The fields points are distributed at the marker points, the centers of the active cells and the face centers of the active cells along the boundary of the computational domain. The field points, at the face centers of the active cells along the boundary of the computational domain, can be also regarded as marker points. They are distinguished from the prescribed marker points and are generated after the grid refinement procedure to represent the boundary of the computational domain.

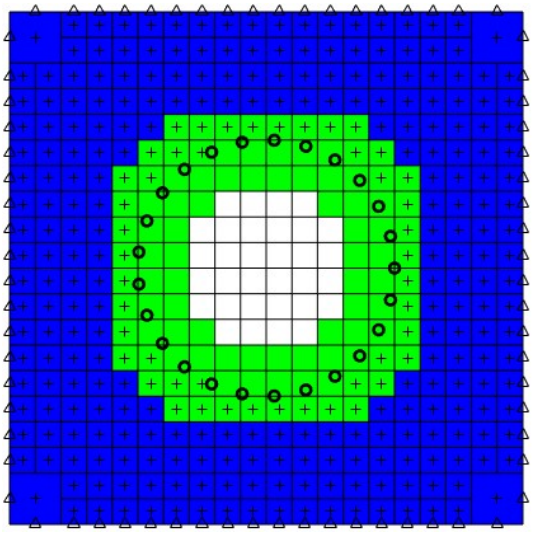


Fig. 14. Field points and cell group. The fields points are represented by the symbols: “circle” coincides with the marker point; “cross” coincides with the center of the active cells; “triangle” coincides with the face center of the active cells along the boundary of the computational domain. The green color corresponds to the interfacial group, the blue color the inner group, and the white color the outer group.

To discretize the Laplace equation at the field point coinciding with the center of an active cell C, a standard HPC can be constructed if all neighboring cells are active cells and the number of the neighboring leaves is equal to eight. Otherwise, we construct an irregular cell to discretize the Laplace equation. The irregular cell is also used for the discretization of the Neumann boundary condition. To guarantee that there are sufficient field points for the harmonic polynomial method, the search of the field points is not limited in the neighboring leaves. We can extend the search region from the neighboring leaves to the leaf cells with $distance\big(cell(*), cell(C)\big) \leq P$, where P is a prescribed parameter and is commonly $2 \sim 3$.

5. Free-surface tracking

The theories of ocean waves [1] and sea loads [4] often assume that the sea water is incompressible and invisicid and the fluid motion is irrotational. A velocity potential φ, satisfying the Laplace equation, can be used to describe the fluid velocity vector $\mathbf{u} = \nabla\varphi$. The water pressure p follows Bernoulli’s equation. The kinematic free-surface condition is that a fluid particle on the free surface always stays on the free surface [4]. In the Cartesian coordinates system with the x-axis along the mean free-surface level and the y-axis positive upwards, the kinematic free-surface condition can be expressed as

$$\frac{\partial \zeta}{\partial t} + \frac{\partial \varphi}{\partial x}\frac{\partial \zeta}{\partial x} - \frac{\partial \varphi}{\partial y} = 0 \quad \text{on} \quad y = \zeta(x,t), \tag{28}$$

which tracks the free-surface elevation $\zeta(x,t)$. The dynamic free-surface condition is simply that the water pressure is equal to the constant atmospheric pressure on the free surface

$$g\zeta + \frac{\partial \varphi}{\partial t} + \frac{1}{2}\left[\left(\frac{\partial \varphi}{\partial x}\right)^2 + \left(\frac{\partial \varphi}{\partial y}\right)^2\right] = 0 \text{ on } \quad y = \zeta(x,t). \tag{29}$$

The free-surface conditions (28) and (29) are nonlinear. In many cases, we are able to simplify the problem and still get sufficient information by linearizing the free-surface conditions

$$\frac{\partial \zeta}{\partial t} = \frac{\partial \varphi}{\partial y} \quad \text{on} \quad y = 0, \tag{30}$$

$$g\zeta + \frac{\partial \varphi}{\partial t} = 0 \text{ on } y = 0. \tag{31}$$

For the fully-nonlinear flow and in condition that the free-surface elevation is single-valued, the free-surface evolution can be tracked in a semi-Lagrangian manner, i.e., by tracking the wave elevation and the corresponding velocity potential through [24]

$$\frac{\partial \zeta}{\partial t} = \frac{\partial \varphi}{\partial y} - \frac{\partial \varphi}{\partial x}\frac{\partial \zeta}{\partial x}\ , \tag{32}$$

$$\frac{D\varphi}{Dt} \coloneqq \frac{\partial \varphi}{\partial t} + \frac{\partial \zeta}{\partial t}\frac{\partial \varphi}{\partial y} = \frac{\partial \zeta}{\partial t}\frac{\partial \varphi}{\partial y} - g\zeta - \frac{1}{2}\left[(\frac{\partial \varphi}{\partial x})^2 + (\frac{\partial \varphi}{\partial y})^2\right]. \tag{33}$$

When the free-surface elevation is multivalued (such as overturning waves), it becomes difficult to apply the semi-Lagrangian method for the evolution of the free surface. A practical way is to adopt the Lagrangian method, i.e., following the water particles on the free surfaces through Eqs. (2)-(3). Eqs. (2)-(3), (30)-(31) and (32)-(33) can be solved by Runge-Kutta methods, e.g. the second-order explicit Runge-Kutta method. It is noted that, the Laplace equation should be solved at each sub step of the Runge-Kutta method for the evaluation of the velocities on the free-surface boundary. Then, the normal velocity on the boundary, $\partial\varphi/\partial n$, can be evaluated by the method presented in Section 3.2 and the tangential velocity, $\partial\varphi/\partial s$, can be evaluated by a local polynomial approximation on the boundary, e.g., $\varphi = a + b \cdot s + c \cdot s^2$. Here, s is the local arc-length coordinate. For the semi-Lagrangian method, the so called 'saw tooth' instability may occur on the free-surface boundary. It can be removed by smoothing techniques [27, 30]. For the Lagrangian method, the fluid particle on the boundary can become too close to each other or too far away from each other, which result in poor results or even numerical instability. To avoid this, the boundary should be re-gridded every time step or after a fixed number of steps [30-33].

6. Numerical examples

6.1 Sloshing in a rectangular tank

The sloshing in a mobile rectangular tank is simulated to examine the robustness and accuracy of the present harmonic polynomial method. The mean water depth *h* and the tank breadth *l* are chosen as *h*=833 mm and *l*=1000 mm, which correspond to Abrahamsen's experiment [34]. The tank is forced to oscillate in the horizontal direction

$$\eta(t) = \sum_{i=1}^{N} \eta_i(t). \tag{34}$$

Here,

$$\eta_i(t) = \begin{cases} \eta_{ai}[\cos(\sigma_i\{t - t_{si}\}) - 1] & t \geq t_{si} \\ 0 & t < t_{si} \end{cases}, \tag{35}$$

where, $\sigma_i = \sqrt{\frac{g\pi i}{l}\tanh\frac{h\pi i}{l}}$, $t_{s1} = 3.820\ s$, $t_{s5} = 0.5652\ s$, $t_{s9} = 0.0\ s$, $\eta_{a1} = 0.0202\ m$, $\eta_{a5} = 0.00145\ m$, and $\eta_{a9} = 0.00077\ m$. The analytical excitation signal was implemented in Abrahamsen's experiment and can be considered as an approximation of the measured tank motion. Fig. 15 shows the measured and analytical data of the tank motion.

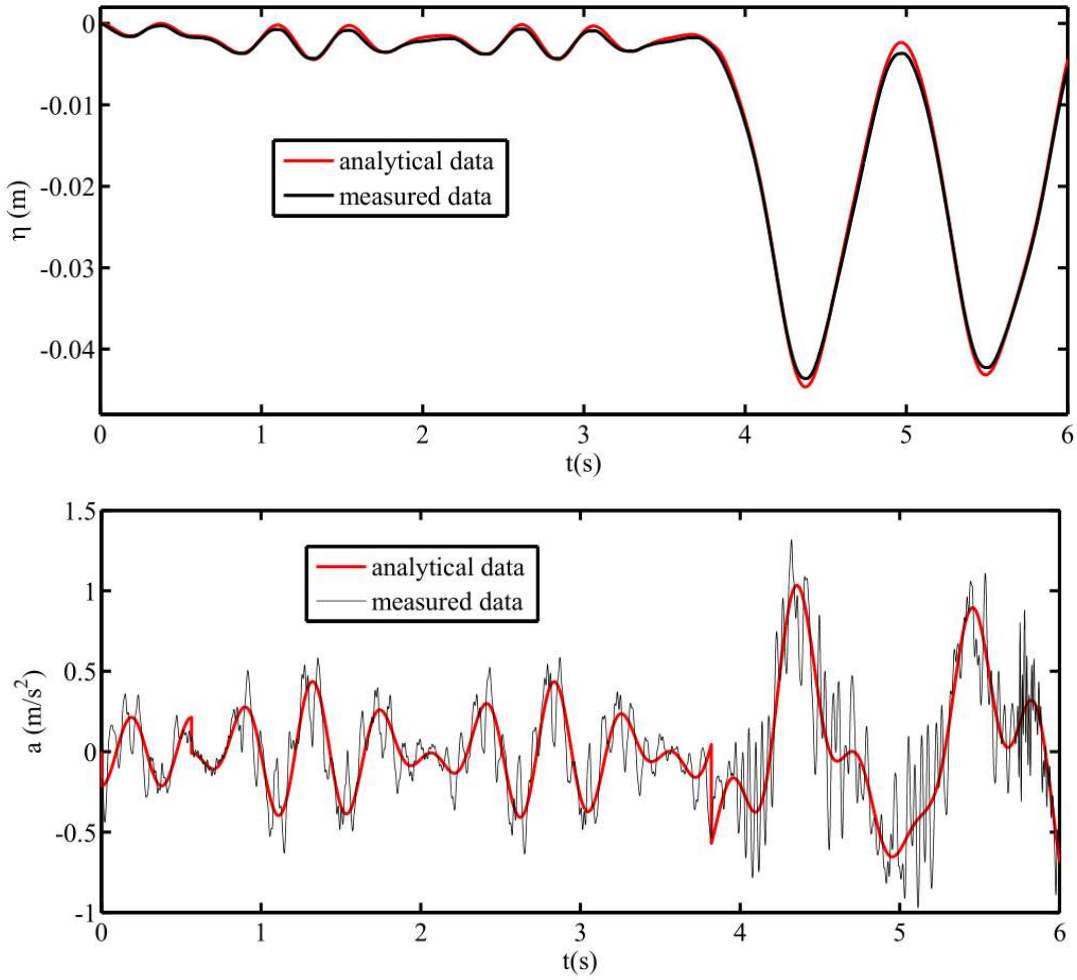


Fig. 15. Measured and analytical data of the horizontal displacement and acceleration of the tank.

For convenience, the problem is solved in the tank-fixed coordinate system, with the origin in the mean free surface at the center plane of the tank, illustrated in Fig. 16.

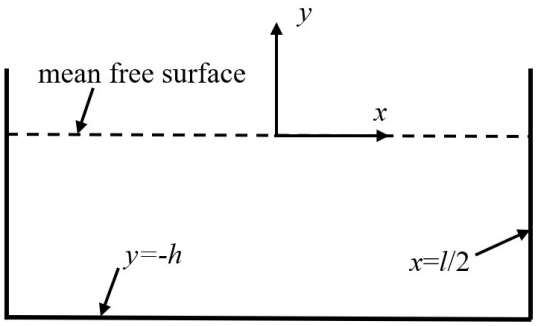


Fig. 16. Coordinate system of the partially-filled rectangular tank.

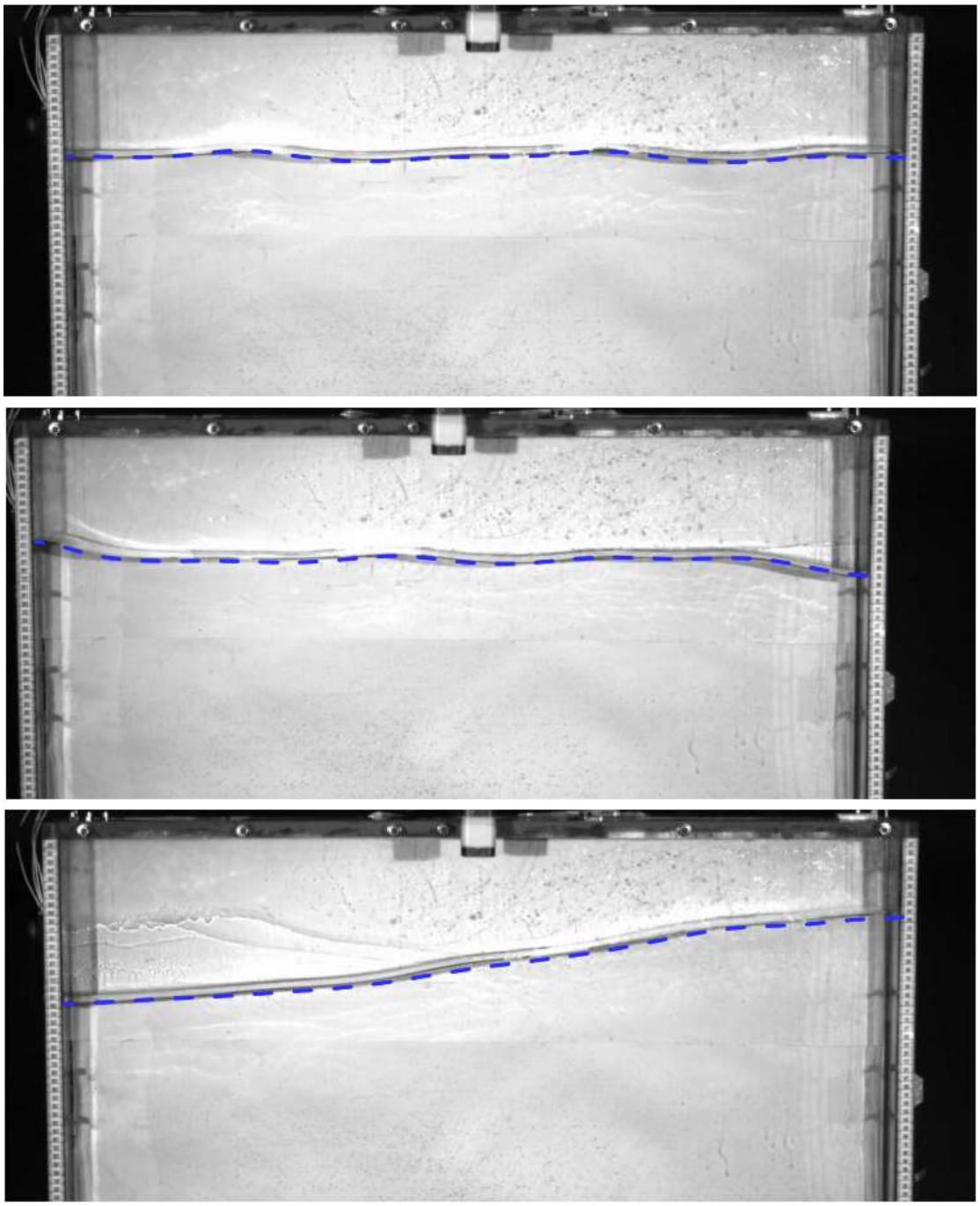
Then, the fully nonlinear free-surface conditions (28)-(29) are modified as (referring to [7], pp 47-49)

$$\frac{\partial \zeta}{\partial t} = \frac{\partial \varphi}{\partial y} - \left(\frac{\partial \varphi}{\partial x} - v_0\right)\frac{\partial \zeta}{\partial x}\ , \tag{36}$$

$$\frac{\partial \varphi}{\partial t} = -g\zeta - \frac{1}{2}\left[\left(\frac{\partial \varphi}{\partial x}\right)^2 + \left(\frac{\partial \varphi}{\partial y}\right)^2\right] + v_0\frac{\partial \varphi}{\partial x}. \tag{37}$$

Here, $v_0 = \dot{\eta}(t)$ is the horizontal moving speed of the tank. We can evaluate v_0 by taking the

numerical differentiation of the measured $\eta(t)$ and use it for the numerical simulation. Fig. 17 compares the numerical free surface with the experimental record, which shows good agreement.



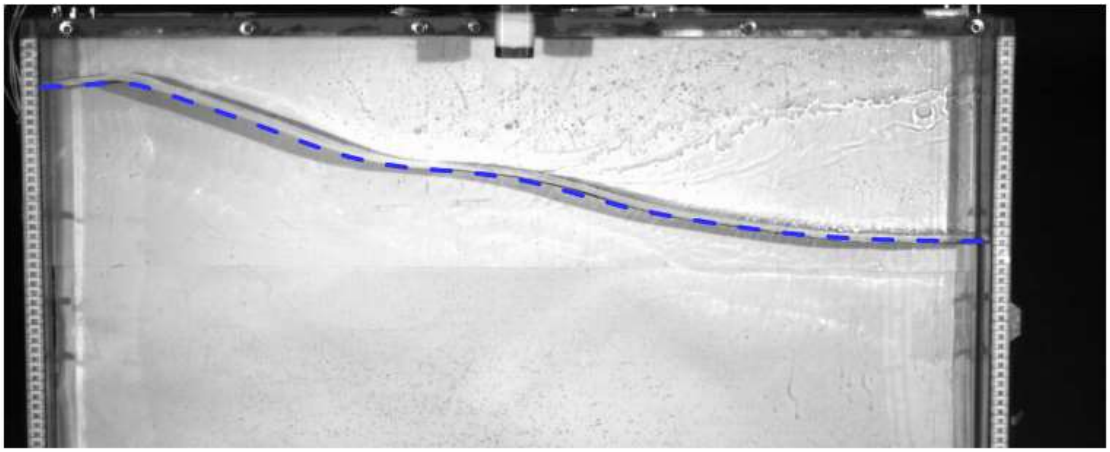


Fig. 17. Comparison between the numerical free surface and the experimental record. The blue dashed lines denote the numerical free surface.

For in-depth comparison, we try to seek the analytical solution of the sloshing problem. The nonlinear sloshing can be represented by the multimodal solution [35]

$$\eta(x,t) = \sum_{j=1}^{\infty} \beta_j(t) f_j(x) \quad \text{with} \quad f_j(x) = \cos[\frac{\pi j}{l}(x + l/2)] \tag{38}$$

$$\varphi(x,y) = v_0 x + \sum_{j=1}^{\infty} R_j(t)\varphi_j(x,y) \quad \text{with} \quad \varphi_j(x,y) = \cos[\frac{\pi j}{l}(x + l/2)] \frac{\cosh[\frac{\pi j}{l}(y+h)]}{\cosh[\frac{\pi j}{l}h]} \tag{39}$$

In practice, we adopt a finite number of modes to approximate the nonlinear sloshing, i.e.,

$$\eta(x,t) \approx \sum_{j=1}^{N} \beta_j(t) f_j(x) \tag{40}$$

$$\varphi(x,y) \approx v_0 x + \sum_{j=1}^{N} R_j(t)\varphi_j(x,y) \tag{41}$$

Inserting (40)-(41) into Eqs. (36)-(37) and making simplification, we obtain

$$\frac{\partial \beta_j}{\partial t} = c_j \ , \ j = 1, 2, \dots, N \tag{42}$$

$$\left(\frac{\partial R_j}{\partial t}\right) = A^{-1} b \ , \tag{43}$$

Where $c_j = \frac{2}{l}\int_{-l/2}^{l/2}[\frac{\partial \varphi}{\partial z} - (\frac{\partial \varphi}{\partial x} - v_0)\frac{\partial \eta}{\partial x}]\cos[\frac{\pi j}{l}(x + l/2)]dx$, $A(i,j) = \int_{-l/2}^{l/2} \varphi_j \cos[\frac{\pi i}{l}(x + l/2)]dx$ and $b(i) = \int_{-l/2}^{l/2}[-\dot{v}_0 x - \frac{1}{2}|\nabla\varphi|^2 - gz + v_0\frac{\partial \varphi}{\partial x}]\cos[\frac{\pi i}{l}(x + l/2)]dx$. Eqs. (42)-(43) can be solved numerically, e.g. by Runge-Kutta methods. The results are referred to as semi-analytical solutions. Because the measured $\dot{v}_0$ has high-frequency noises, it could be better to adopt the analytical tank motion for the comparison between the semi-analytical solution and the numerical solution. In the present calculation, the semi-analytical model adopts the modes up to $N = 10$ and the second-order Runge-Kutta method with the time step size $\Delta t = 0.0002\ s$. The harmonic polynomial model is implemented on the Cartesian grid of 60×60 square elements, representing the computational domain of $1m \times 1m$. The free-surface boundary is represented by the markers with a fixed spacing, Δx, in the x direction. The wetted surface of the two vertical walls are also distributed by equally-spaced markers with the spacing equal to Δx. The evolution of the free surface is tracked in the semi-Lagrangian manner by the second-order Runge-Kutta method. The grid system is re-generated every time step. The third-order harmonic polynomials are used to construct the irregular cells. It is found that $\Delta x = 1/240$ is fine enough in space and $\Delta t = 0.0002\ s$ is sufficiently small. Fig. 18 presents the five lowest modes of the wave elevation

based on the analytical tank motion. It can be seen that the present solution by the harmonic polynomial method agrees well with the semi-analytical solution.

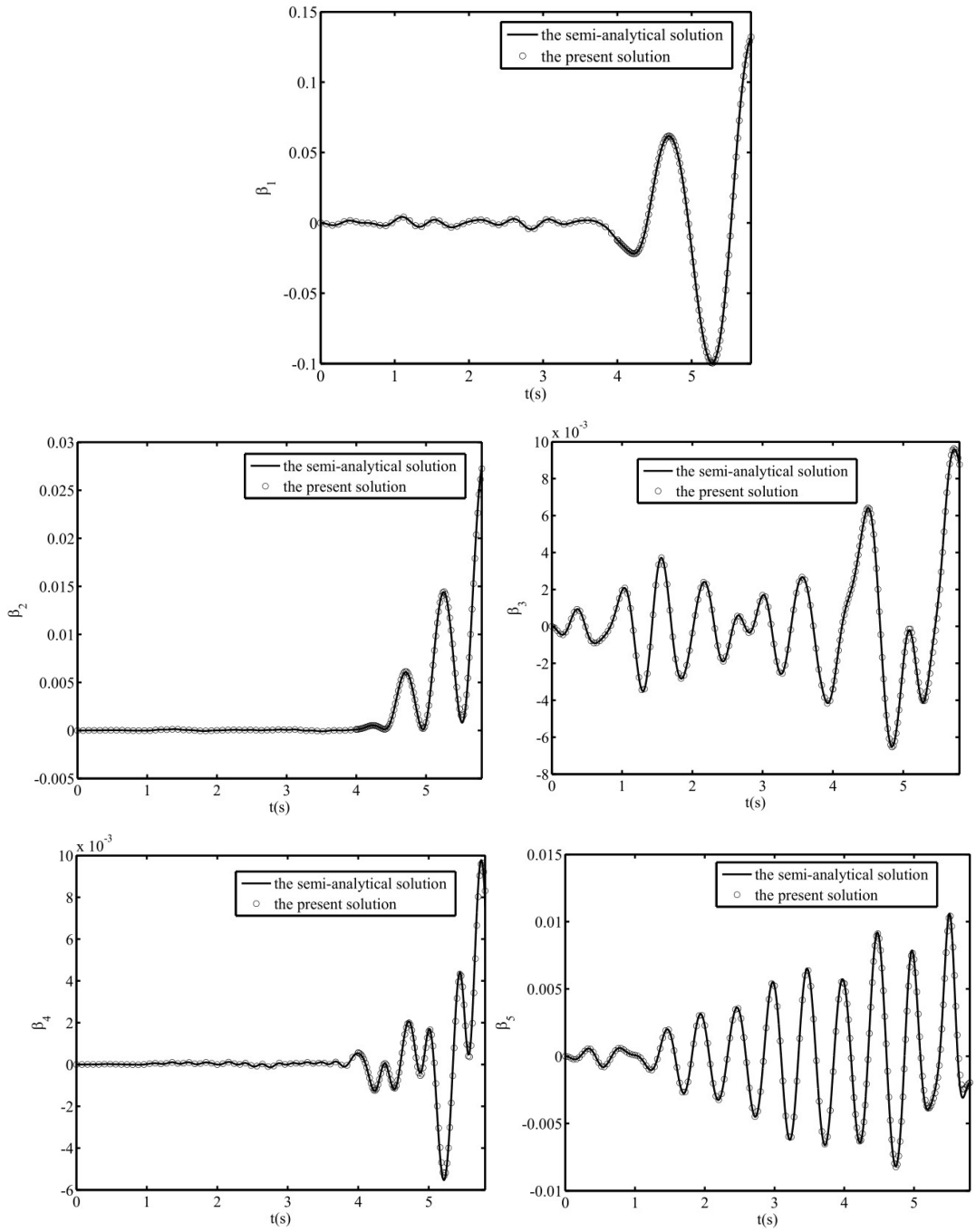


Fig. 18. Comparison of the five lowest modes of the wave elevation between the numerical solution and the semi-analytical solution.

6.2 Water entry of wedges

When a wedge enters the water surface, the water rises up and jets are formed on the wetted body surface [36-38]. The jet flow can be thin and long for small deadrise angles. For the deadrise angle less than 45°, the pressure on the wetted body surface is characterized by a peak near the root of the jet flow. All these result in numerical challenges especially for small deadrise angles. By neglecting the gravity, Dobrovol'skaya [36] presented the similarity solutions, which analytically represent wedges entering the

water surface with constant velocity. The similarity solutions are implicitly given and require solving a nonlinear singular integral equation. Zhao & Faltinsen [37] presented good results for deadrise angles equal to and less than 30°. More accurate results were given by Wang & Faltinsen [38]. In the present study, the proposed harmonic polynomial method is employed to simulate the water entry problem. The numerical results are compared to the similarity solutions to verify the robustness and accuracy of the present method. Since the flow is symmetrical about the center plane of the wedge, the problem is solved only in half of the fluid domain, illustrated in Fig. 19. The truncation boundary S_I and the bottom boundary S_B are introduced to reduce the computational effort. They are far away from the wedge, giving negligible influence on the water entry process. The Cartesian grid with local refinement is adopted in the numerical calculation. Equally-spaced markers are distributed along the wetted body surface. In the region close to the wedge body, the spacing of the markers on the free surface is equal to that of the markers on the wedge surface. Far away from the wedge, the larger spacing is adopted. The grid system is illustrated in Fig. 20.

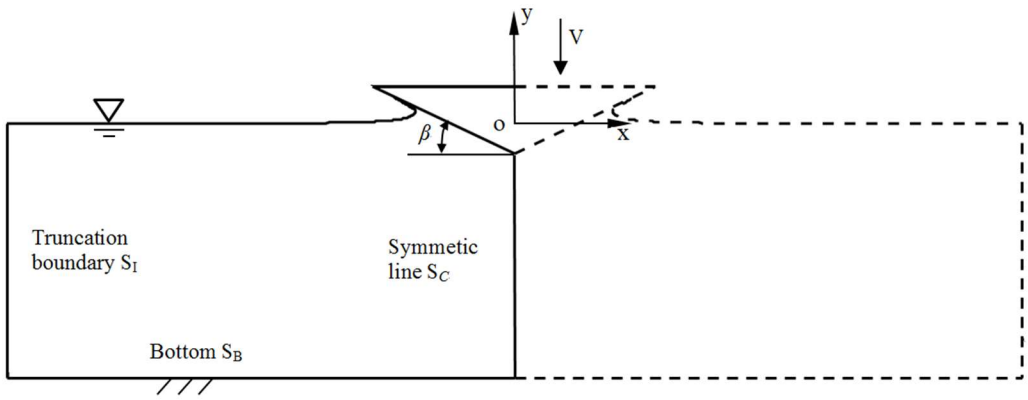
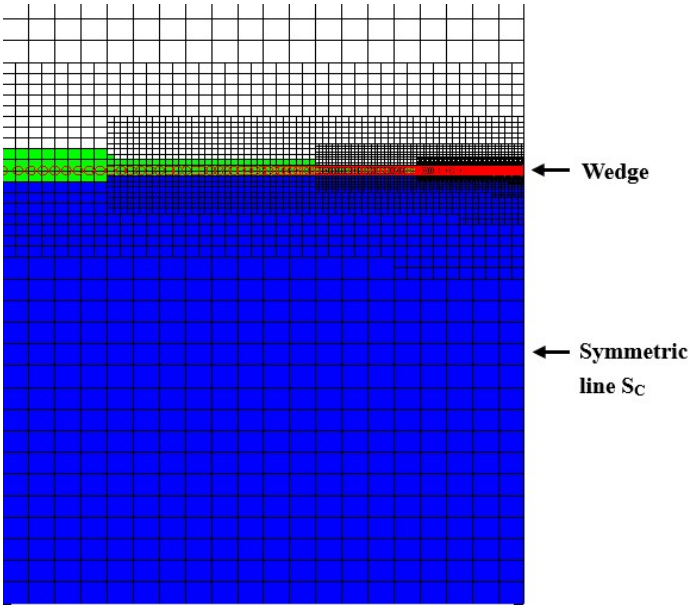
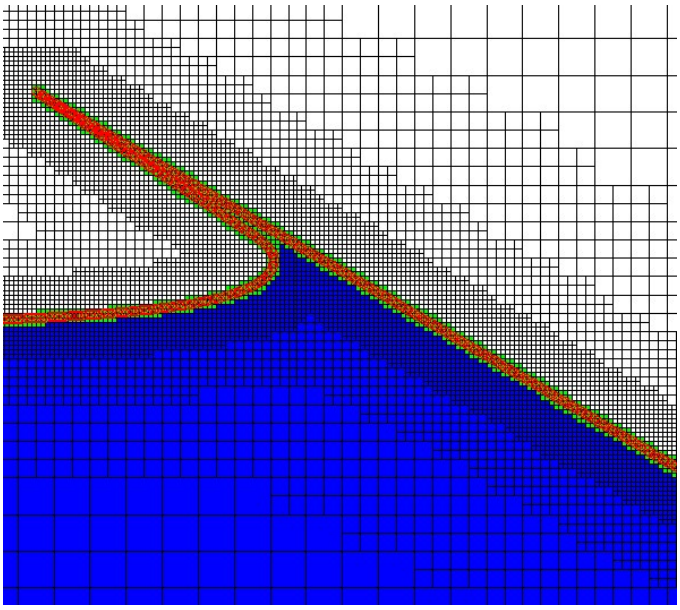


Fig. 19. Coordinate system and sketch of a wedge vertically entering into calm water.



(a) Global view



(b) Local view near the wedge body

Fig. 20. Illustration of the grid system for the water entry of a wedge. The deadrise angle of the wedge is 30°. The red circles denote the markers on the boundary.

Based on the Bernoulli's equation, the pressure on the body is expressed as

$$p = -\rho(gy + \frac{\partial \varphi}{\partial t} + \frac{1}{2}|\nabla\varphi|^2). \tag{44}$$

The $\partial\varphi/\partial t$ term can be evaluated by solving the boundary value problem for the auxiliary function $\psi = \partial\varphi/\partial t + \mathbf{V}\cdot\nabla\varphi$, which satisfies [39]

$$\nabla^2\psi = 0. \tag{45}$$

The Bernoulli's equation gives the Dirichlet boundary condition for ψ on the free surface

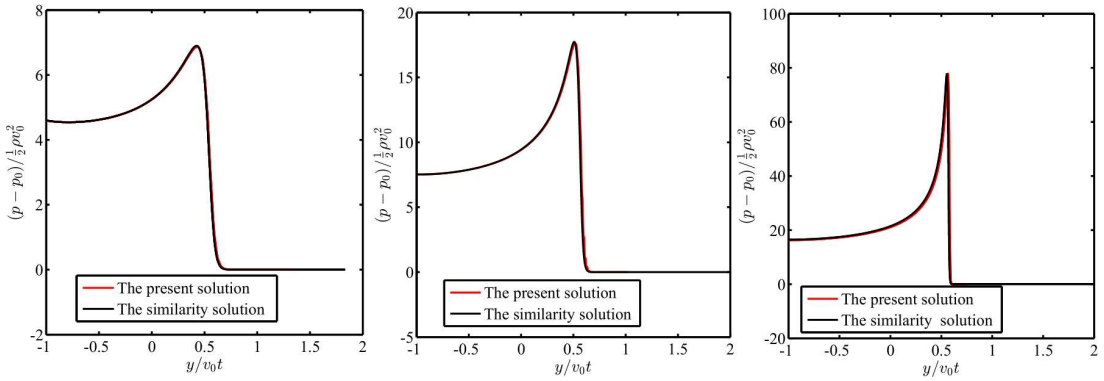
$$\psi = \mathbf{V}\cdot\nabla\varphi - \frac{1}{2}|\nabla\varphi|^2 - gy. \tag{46}$$

For constant entry speed, the boundary condition for ψ can be derived as

$$\frac{\partial\psi}{\partial n} = 0. \tag{47}$$

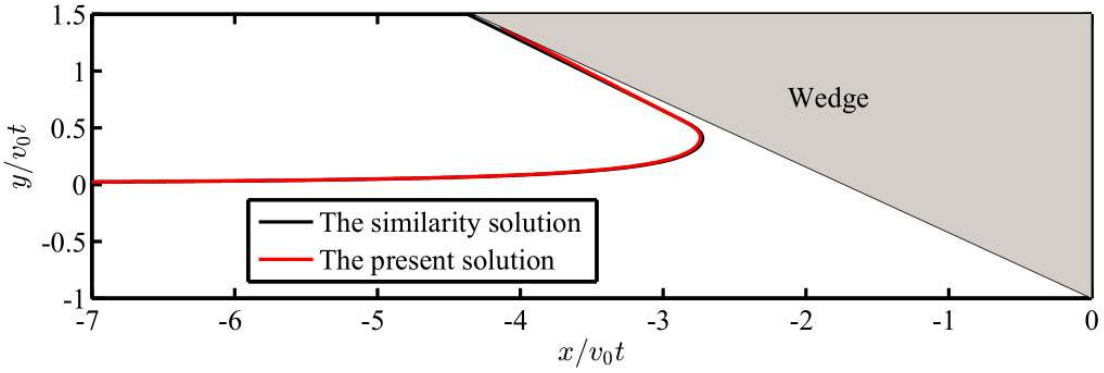
Far away from the body, ψ should vanish

$$\psi = 0. \tag{48}$$

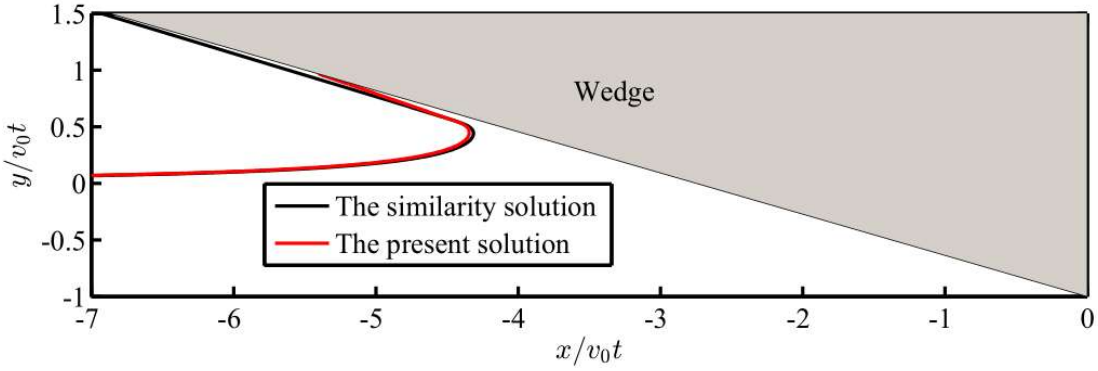


(a) $\beta = 30°$ (b) $\beta = 20°$ (c) $\beta = 10°$

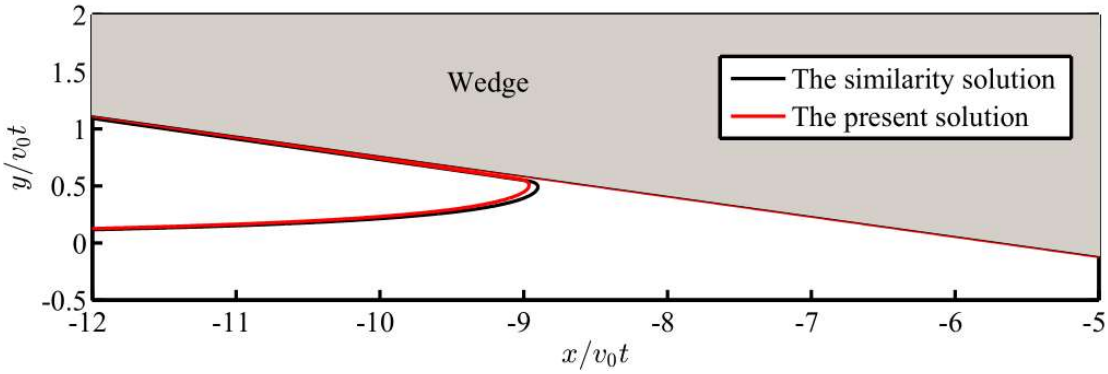
Fig. 21. Comparison of the pressure distribution on the wetted wedge surface for wedges with deadrise angles of 10°, 20°and 30°. v_0 denotes the water entry speed of the wedge.



(a) $\beta = 30°$



(b) $\beta = 20°$

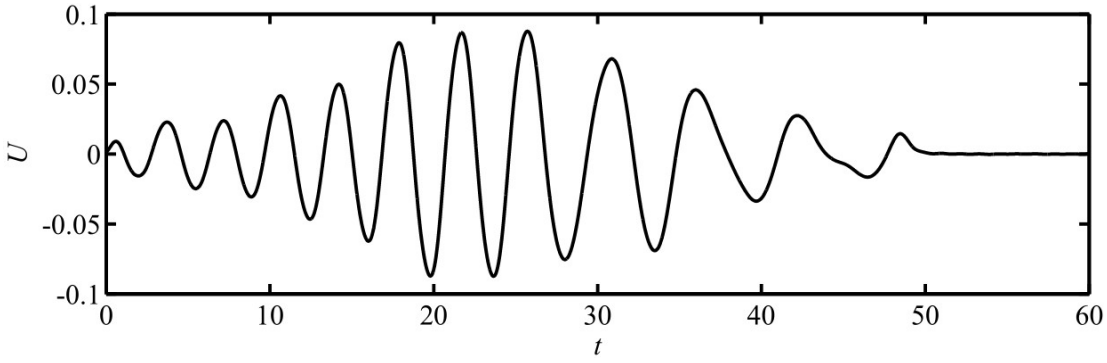


(c) $\beta = 10°$

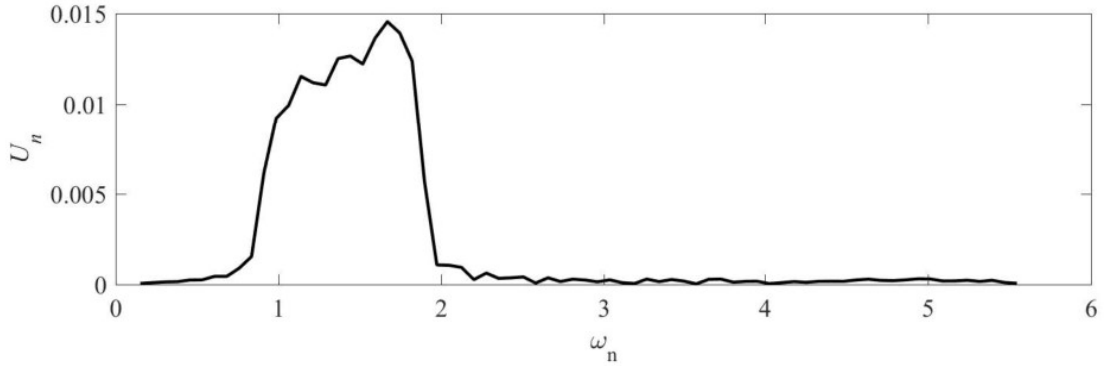
Fig. 22. Comparison of the free surface elevation near the wedge. v_0 denotes the water entry speed of the wedge.

6.3 Deep-water plunging breakers

Dommermuth et al. [32] produced a very steep and eventually plunging wave experimentally by generating a wave packet with a programmed piston-type wavemaker. The experiments took place in a 25 m long, 0.7 m wide, and 0.6 m deep water channel. The tank was fitted with a damping beach starting at a distance 19.5 m from the wavemaker. Dommermuth et al. adopted the water depth h as the reference length and $\sqrt{gh}$ the reference velocity. They presented the measured time history of the wave maker velocity $U(t)$ as a non-dimensional Fourier-cosine series, $U(t) = \sum_{n=1}^{72} U_n \cos(\omega_n t - \theta_n)$, which is shown in Fig. 23.



(a) Time-domain representation



(b) Frequency-domain representation

Fig. 23. The measured wave maker velocity. The time, frequency and velocity are presented in dimensionless form.

In the numerical simulation, a 30 m long numerical wave tank is adopted, which is illustrated in Fig. 24. The fully-Lagrangian method, solving Eqs. (2)-(3), is used to track the evolution of the free surface. In the numerical damping zone, the free surface conditions are modified as

$$\frac{Dy}{Dt} = \frac{\partial \varphi}{\partial y} - \alpha y, \text{ on } S_F \tag{49}$$

$$\frac{D\varphi}{Dt} = \frac{1}{2}|\nabla\varphi|^2 - g\zeta - \alpha\varphi, \text{ on } S_F \tag{50}$$

It is reasonable to choose the damping coefficient as follows

$$\alpha = \begin{cases} 1.4\sqrt{\frac{g}{h}}\left[1 - \cos\left(\frac{\pi\xi}{2}\right)\right]/2, & \xi \le 2 \\ 1.4\sqrt{\frac{g}{h}}, & \xi > 2 \end{cases}, \tag{51}$$

since the wave has a central frequency of about $1.4\sqrt{g/h}$ (see Fig. 23 (b)) and the corresponding wave length is about 2 m. The harmonic polynomial model is implemented on the Cartesian grid of 310×10 square elements, representing the computational domain of $31m \times 1m$. The mean position of the piston is set to be 1 m away from the left boundary of the computational domain. The boundary of the water domain is represented by markers. Along the wetted piston surface, 81 markers are equally distributed. Along the free surface, the spacing of the markers is limited up to $h/80$; smaller spacing should be adopted at the highly-curved region, e.g. the plunging breaker. The spacing of the markers controls the local refinement of the Cartesian grid, as detailed in Section 4. The markers are redistributed along the boundary every time step. It is the so called 're-gridding' procedure, mentioned in Section 5, to avoid

numerical instability. On the free surface, we can represent the markers in the functions of the chord-length parameter s: $x(s), y(s)$ and $\varphi(s)$. The free surface boundary is regarded to be a line segment in a one-dimensional space with the coordinate s. It is easy to evaluate the curvature of the boundary $\kappa(s)$. The line segment can be represented by a one-dimensional Cartesian grid system with the element size approximately equal to $h/80$. We adopt Eq. (26), where Δh is set to be $h/80$ and $l = \min(h/80, 0.2/\kappa(s))$, to specify the depth of the markers. Then the one-dimensional Cartesian grid system can be locally refined by the method similar to that presented in Section 4.2. The control parameter N is set to be 10 for slowly-varying grid size. Finally, we redistribute new markers on the one-dimensional locally-refined Cartesian grid system, which is illustrated by Fig. 25. The information of the new markers, i.e., x, y and φ, can be interpolated by the old markers, for instance, by the method presented in [33].

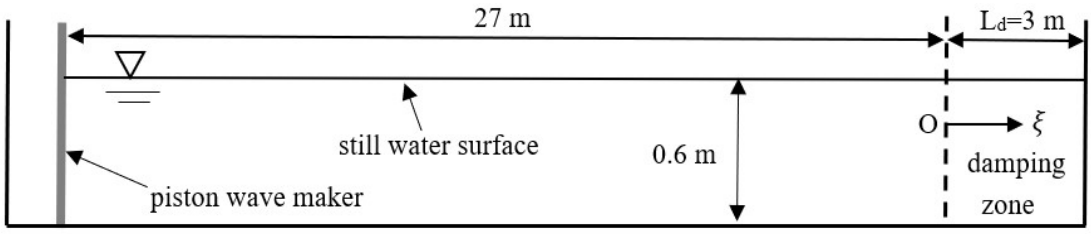


Fig. 24. The sketch of the numerical wave tank.

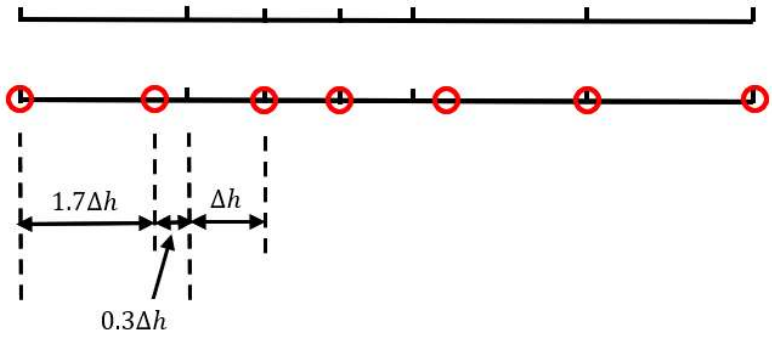
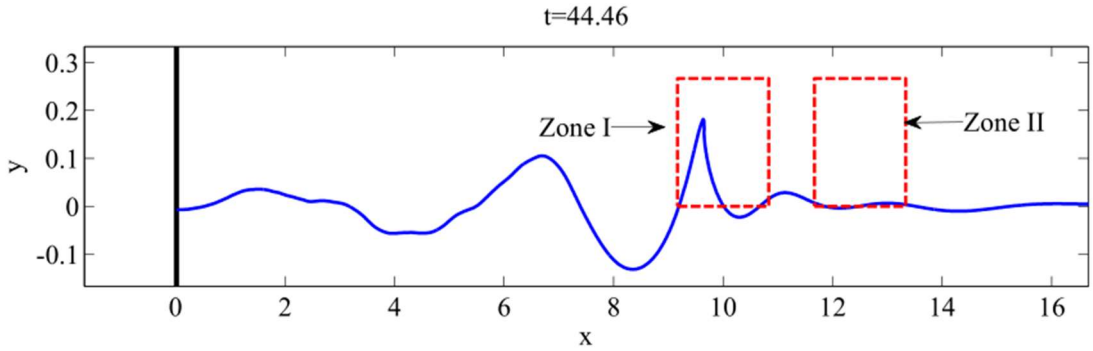
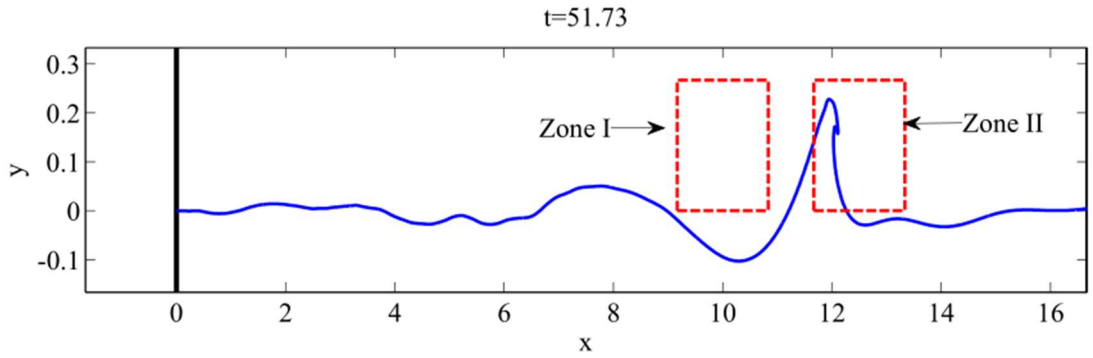


Fig. 25. Distribution of markers on the one-dimensional grid system. The red circles denote the new markers.

Two plunging waves, shown in Fig. 26, are observed in the numerical simulation. Fig. 27 illustrates the second plunging wave and the corresponding grid system shortly before re-entering the free surface. Dommermuth et al. [32] did not report the first plunging wave. The reason for this could be that their grid system (with the grid size of 0.6/25 m) is too coarse to capture the detail of the small overturning part. We tried the coarse grid and neither found it. It notes that the first plunging wave will finally re-enter the free surface, which results in the breakdown of the potential flow solver. To avoid this, the markers, with the radius of the curvature less than 0.01 m, on the free surface, are removed every time step before $t < 48.5$.



(a) First plunging wave



(b) Second plunging wave

Fig. 26. Instantaneous free-surface profile of the simulated plunging waves at t=44.46 and t=51.73.

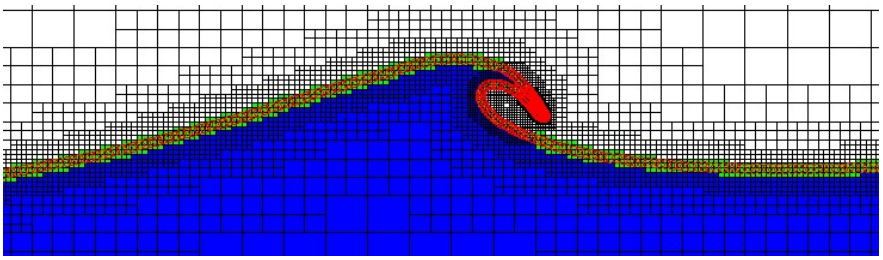
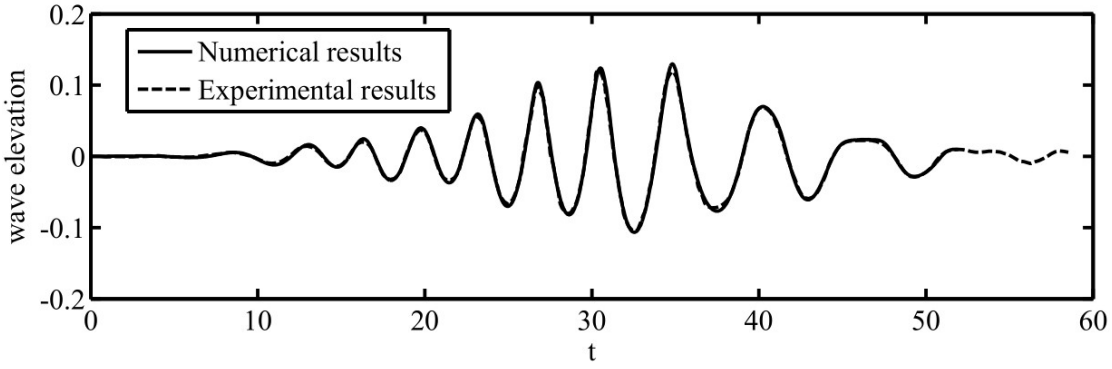
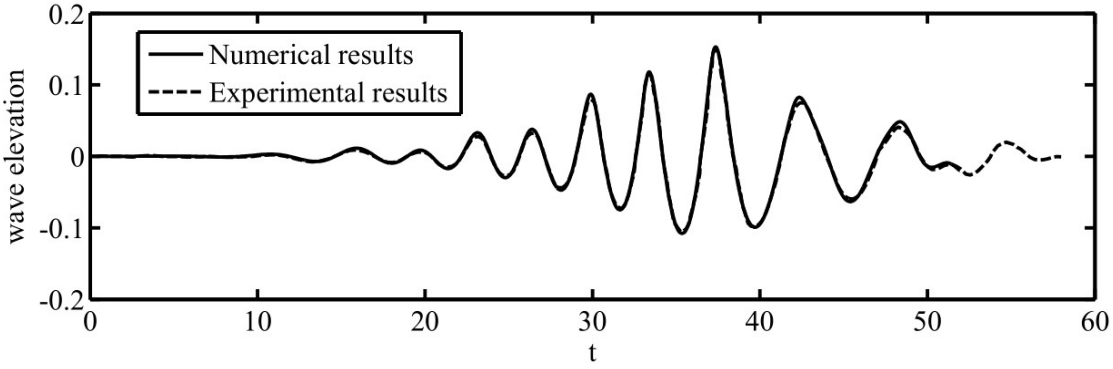


Fig. 27. Numerical free-surface profile of the second plunging wave and the corresponding grid system. The red circles denote the markers on the free-surface boundary.

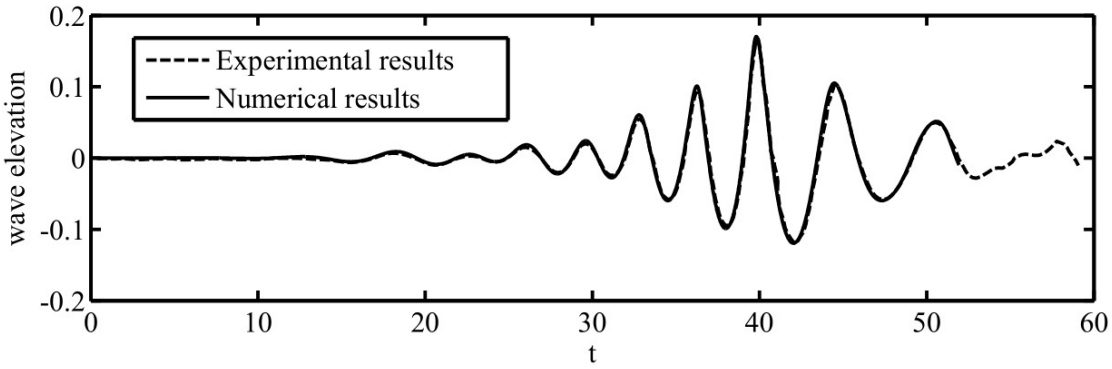
Fig. 28 compares the numerical wave elevation to the experimental data at several locations. In general, good agreement is obtained expect for a short duration at x=10.83, which is marked in the subplot (e). We notice the following facts: x=10.83 is shortly downstream of the first plunging wave; the duration of the discrepancy is shortly after the plunger re-entering the free surface; the numerical wave elevation overestimates the experimental data. Then the reason for the discrepancy could be that the re-entering of the plunger, not interpreted by the present numerical solver, results in the energy dissipation. Fig. 29 compares the numerical water-particle velocities to measurements, giving good agreement.



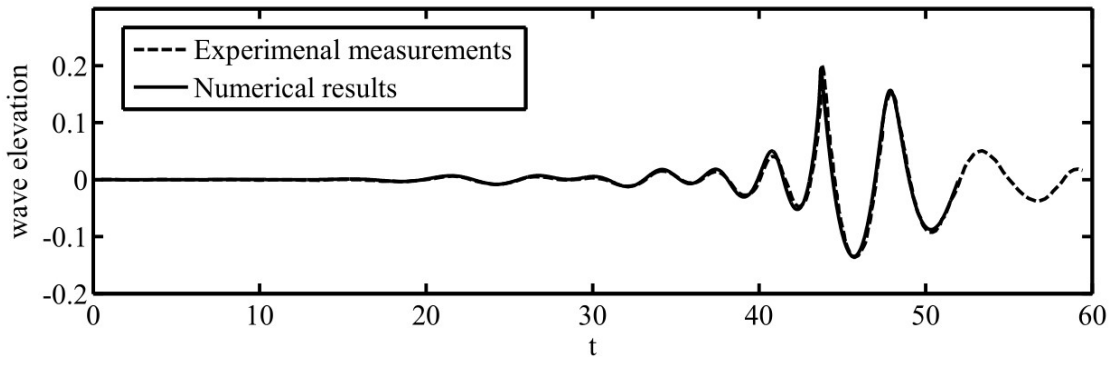
(a) x=3.17



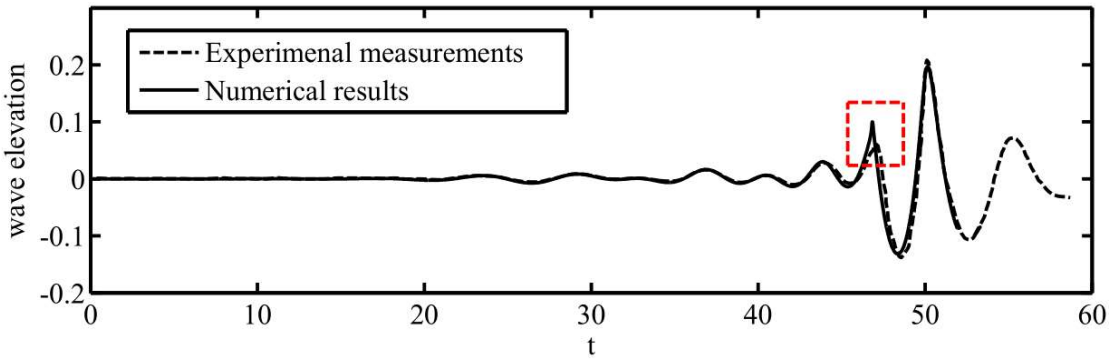
(b) x=5.00



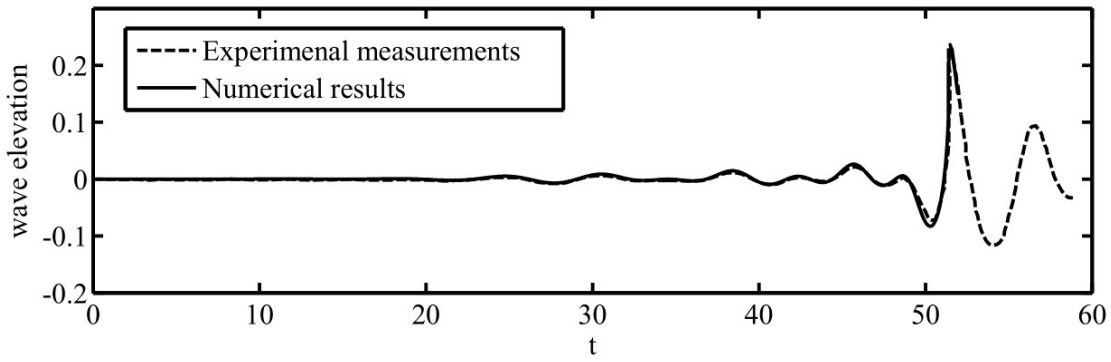
(c) x=6.67



(d) x=9.17

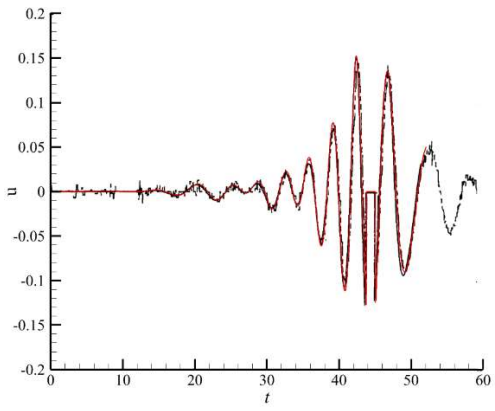


(e) x=10.83

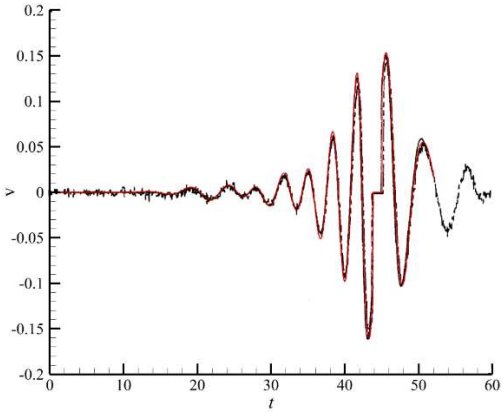


(f) x=11.83

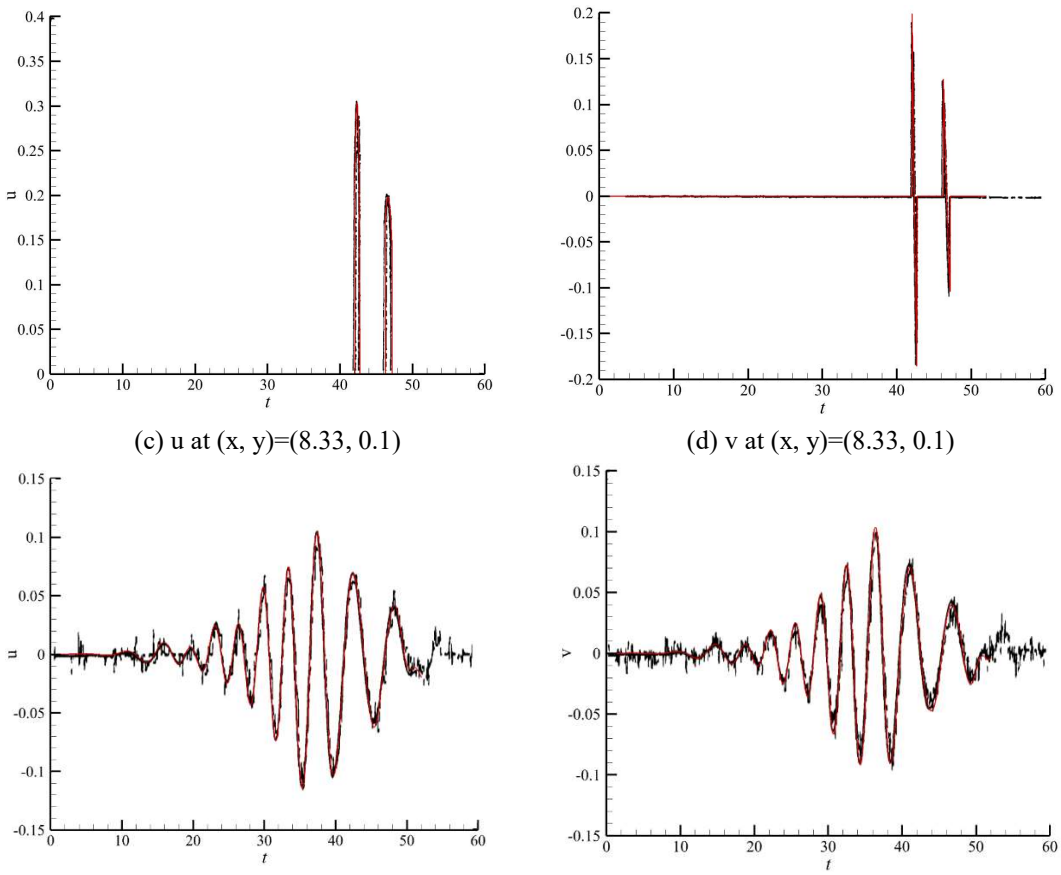
Fig. 28. Numerical free-surface elevations compared to wave-probe measurements as a function of time at distances from the wavemaker of (a) x=3.17, (b) x=5.00, (c) x=6.67, (d) x=9.17, (e) x=10.83 and (f) x=11.83.



(a) u at (x, y)=(8.33, -0.1)



(b) v at (x, y)=(8.33, -0.1)



(c) u at (x, y)=(8.33, 0.1) (d) v at (x, y)=(8.33, 0.1)

(e) u at (x, y)=(5.00, -0.25) (f) v at (x, y)=(5.00, -0.25)

Fig. 29. Numerical water-particle velocities compared to measurements as a function of time. u is the horizontal velocity and v the vertical velocity; the red lines denote the numerical results and the black lines the measurements.

7. Summary

This work developed a high-order harmonic polynomial method for solving the Laplace equation with complex boundaries. The irregular cell was proposed for the accurate discretization of the Laplace equation at the field point, where it is difficult to adopt a high-quality harmonic polynomial cell. For a given field point, the irregular cell is constructed by the neighboring field points. It adopts complete harmonic polynomials up to order k. The choice of k is somewhat arbitrary. The stability may benefit from small k and the accuracy from large k. The authors prefer to use $k = 3$, which balances the accuracy and the stability, for the fully-nonlinear free-surface flows. It notes that the number of the neighboring field points used for the construction of the irregular cell should be sufficiently larger than $2k + 1$ to avoid singularity. Based on the irregular cell, a new high-order scheme has been developed for discretizing the normal derivative of the potential functions on boundaries. The irregular cell and the proposed scheme for the evaluation of the normal derivative deal with the complex boundaries in a simple and accurate manner. They avoid the time-consuming searching procedure for an optimized harmonic polynomial cell [23], the complex procedure of the immersed boundary method and the possible spurious force oscillation [24, 25, 29], and the possible cumbersome procedure for the generation of high-quality boundary-fitted grids. These advantages make the present method competitive in the accurate, efficient and stable modeling of the highly-nonlinear free-surface flows with and without moving bodies. In the

comparative studies between the harmonic polynomial cell and the irregular cell, we have shown that the standard harmonic polynomial cell is very good. To utilize the advantages of the standard harmonic polynomial cells, it is natural to adopt the Cartesian grid. To increase the resolution of the grid system in the interesting region, the local refinement procedure is adopted, resulting in the locally-refined Cartesian grid system. The boundaries are represented by markers, which can be independent of the grid system and where the boundary conditions are satisfied. The depth of markers controls the refinement procedure of the Cartesian grid. Virtual markers can be introduced in the region where the grid is to be refined. The computational cost of the generation of the grid system is negligible compared to building up and solving the matrix system, which is confirmed by our experience of using the adaptive grid (the grid system is re-generated at each time step) to simulate highly-nonlinear free-surface potential flows. It is an advantage compared to boundary-fitted grids, because the generation of high-quality boundary-fitted grids can be a cumbersome and time-consuming task especially for the three-dimensional cases with complex boundaries. With the semi-Lagrangian or fully-Lagrangian free-surface tracking methods, the harmonic polynomial method has been successfully applied to the accurate and stable modelling of the highly-nonlinear free-surface potential flows with and without moving bodies. In the next paper, the proposed method will be generalized to the three-dimensional space. The procedure is not necessarily straightforward, for example, using complete harmonic polynomials up to the order higher than two can be infeasible [26] and an efficient algorithm is required to select the harmonic polynomial basis.

Acknowledgement

This work was mainly supported by the Research Council of Norway through the Centres of Excellence funding scheme AMOS (Project No. 223254). J. Wang and W. Y. Duan also appreciate the financial support from the Ministry of Industry and Information Technology of P. R. China (Numerical Tank Project) and the Harbin Engineering University (Project No. GK2010260309). J. Wang thanks Dr. B. C. Abrahamsen at SINTEF Ocean for providing the experimental data of the sloshing problem.

References

1. Mei CC, Stiassnie MA, Yue DK. Theory and Applications of Ocean Surface Waves. World Scientific Publishing Company; 2005.

2.Dean RG, Dalrymple RA. Water Wave Mechanics for Engineers & Scientists. World Scientific Publishing Company; 1991.

3. Newman JN. Marine Hydrodynamics. The MIT Press; 1977.

4. Faltinsen OM. Sea Loads on Ships and Offshore Structures. Cambridge University Press; 1990.

5. Chorin AJ. Numerical solution of the Navier–Stokes equations. Math. Comput. 1968; 22: 745–762.

6. Bardazzi A, Lugni C, Antuono M, Graziani G, Faltinsen OM. Numerical solution of the Navier–Stokes equations. J. Comput. Phys. 2015; 299: 630–648.

7. Faltinsen OM, Timokha AN. Sloshing. Cambridge University Press; 2009.

8. Faltinsen, OM. Hydrodynamics of High-Speed Marine Vehicles. Cambridge University Press; 2005.

9. Huseby M, Grue J. An experimental investigation of higher-harmonic wave forces on a vertical cylinder. J. Fluid Mech. 2000; 414: 75–103.

10. Kristiansen T, Faltinsen OM. Higher harmonic wave loads on a vertical cylinder in finite water depth. J. Fluid Mech. 2017; 833: 773-805.

11. Fuhrman DR, Madsen PA, Bingham HB. A numerical study of crescent waves. J. Fluid Mech. 2004; 513: 309–341.

12. Bateman WJD, Swan C, Taylor PH. On the efficient numerical simulation of directionally spread surface water waves. J. Comput. Phys. 2001; 174: 277–305.
13. Bingham HB, Zhang H. On the accuracy of finite difference solutions for nonlinear water waves. J. Eng. Math. 2007; 58: 211–228.
14. Engsig-Karup AP, Bingham HB, Lindberg O. An efficient flexible-order model for 3D nonlinear water waves. J. Comput. Phys. 2009; 228: 2100–2118.
15. Engsig-Karup AP, Madsen MG, Glimberg SL. A massively parallel GPU-accelerated model for analysis of fully nonlinear free surface waves. Int. J. Numer. Methods Fluids 2012; 70: 20–36.
16. Ma QW, Wu GX, Eatock Taylor R. Finite element simulation of fully non-linear interaction between vertical cylinders and steep waves. Part 1: Methodology and numerical procedure. Int. J. Numer. Methods Fluids 2001; 36: 265–285.
17. Yan S, Ma QW. Numerical simulation of fully nonlinear interaction between steep waves and 2D floating bodies using the QALE-FEM method. J. Comput. Phys. 2007; 221: 666–692.
18. Rokhlin V. Rapid solution of integral equations of classical potential theory. J. Comput. Phys. 1985; 60: 187–207.
19. Fochesato C, Dias F. A fast method for nonlinear three-dimensional free-surface waves. Proc. R. Soc., Math. Phys. Eng. Sci. 2006; 465: 2715–2735.
20. Wu GX, Eatock Taylor R. Time stepping solutions of the two-dimensional nonlinear wave radiation problem. Ocean Eng. 1995; 22: 785–798.
21. Shao YL, Faltinsen OM. Towards efficient fully-nonlinear potential-flow solvers in marine hydrodynamics. In: Proc. of the 31st Int Conf on Ocean, Offshore and Arc Eng (OMAE), 2012.
22. Liang H, Faltinsen OM, Shao YL. Application of a 2D harmonic polynomial cell (HPC) method to singular flows and lifting problems. Appl Ocean Res. 2015; 53: 75-90.
23. Ma S, Hanssen FW, Siddiqui MA, Greco M, Faltinsen OM. Local and global properties of the harmonic polynomial cell (HPC) method: in-depth analysis in two dimensions. Int J Numer Methods Eng. 2018; 113: 681–718.
24. Hanssen FW, Bardazzi A, Lugni C, Greco M. Free-surface tracking in 2D with the harmonic polynomial cell method: Two alternative strategies. Int J Numer Methods Eng. 2018; 113: 311–351.
25. Hanssen FW, Greco M, Shao YL. The harmonic polynomial cell method for moving bodies immersed in a cartesian background grid. In: Proc. of the 34th Int Conf on Ocean, Offshore and Arc Eng (OMAE), 2015.
26. Shao YL, Faltinsen OM. A harmonic polynomial cell (HPC) method for 3D Laplace equation with application in marine hydrodynamics. J. Comput. Phys. 2014; 274: 312–332.
27. Longuet-Higgins MS, Cokelet ED. The deformation of steep waves on water I. A numerical method of computation. Proc. R. Soc. Lond. A. 1976; 350: 1-26.
28. Peskin CS. Flow patterns around heart valves: a numerical method. J. Comput. Phys. 1972; 10: 252–271.
29. Lee J, Kim J, Choi H, Yang KS. Source of spurious force oscillations from an immersed boundary method for moving-body problems. J. Comput. Phys. 2011; 230: 2677–2695.
30. Sun H. A boundary element method applied to strongly nonlinear wave-body interaction problems, PhD thesis, Norwegian University of Science and Technology, 2007.
31. Dommermuth DG, Yue DK. Numerical simulations of non-linear axisymmetric flows with a free surface. J. Fluid Mech. 1987; 178: 195–219.
32. Dommermuth DG, Yue DK, Lin WM, Rapp RJ, Chan ES, Melville WK. Deep-water plunging

breakers: a comparison between potential theory and experiments. J Fluid Mech. 1988; 189: 423-442.
33. Wang J, Faltinsen OM. Numerical investigation for air cavity formation during the high-speed water entry of wedges, J. Offshore Mech. Arct. Eng 2013; 135: 011101.
34. Abrahamsen BC, Faltinsen OM. The effect of air leakage and heat exchange on the decay of entrapped air pocket slamming oscillations. Phys. Fluids 2011; 23: 102107.
35. Faltinsen OM, Rognebakke OF, Lukovsky IA, Timokha AN. Multidimensional modal analysis of nonlinear sloshing in a rectangular tank with finite water depth. J. Fluid Mech. 2000; 407: 201–234.
36. Dobrovol'skaya ZN. On some problems of similarity flow of fluid with a free surface. J. Fluid Mech. 1969; 36: 805–829.
37. Zhao R, Faltinsen OM. Water entry of two-dimensional bodies. J. Fluid Mech. 1993; 246: 593–612.
38.Wang J, Faltinsen OM. Improved numerical solution of Dobrovol'skaya's boundary integral equations on similarity flow for uniform symmetrical entry of wedges. Appl. Ocean Res. 2017; 66: 23–31.
39. Wang J, Faltinsen OM, Lugni C. Unsteady hydrodynamic forces of solid objects vertically entering the water surface. Phys. Fluids 2019; 31: 027101.