Local and Global Properties of the Harmonic Polynomial Cell Method: In-depth Analysis in Two Dimensions

S. Ma[1,*,†], F-C.W. Hanssen[1], M.A. Siddiqui[1], M. Greco[1,2], O.M. Faltinsen[1]

[1] *Centre for Autonomous Marine Operations and Systems (AMOS), Department of Marine Technology, NTNU, Trondheim, Norway*
[2] *CNR-INSEAN, National Research Council-Marine Technology Institute, Via di Vallerano 139, 00128 Roma, Italy*

Abstract

A detailed and systematic analysis is performed on the local and global properties of the recently developed Harmonic Polynomial Cell (HPC) method, a very accurate and efficient field solver for problems governed by the Laplace equation. At the local cell level, a simple rule is identified for the proper choice of harmonic polynomials in the local representation of the velocity potential in cells with symmetry properties. The local solution error, its convergence rate, its dependence on the cell topology, its distribution inside the cell and its features across cells with different dimensions, are carefully examined with relevant findings for HPC numerical implementations. At the global level, the error convergence rate is analytically estimated in terms of error contributions from the boundary conditions and from inside the liquid domain. In most cases, the error associated with boundary conditions dominates the global error. In order to minimize it, Quadtree grid strategies or high-order local expressions of the velocity potential are proposed for cells near critical boundary portions. To model accurately the boundary conditions on rigid or deformable surfaces with generic geometries, three different grid strategies are proposed by adopting concepts of immersed boundary method and overlapping grids. They are comparatively studied for a circular rigid cylinder in infinite fluid and for the propagation of a free-surface wave. Then, an immersed boundary strategy, using numerical choices suggested in this paper, is successfully compared against a fully nonlinear Boundary Element Method for the case of a surface-piercing circular cylinder heaving in otherwise calm water.

Keywords

Harmonic polynomial cell method
Laplace equation
Grid convergence study
Quadtree grid
Immersed boundary method
Overlapping grid

List of relevant abbreviations

2D	Two Dimensional	GS	Global Solution
3D	There Dimensional	HP	Harmonic Polynomial
B.C.	Boundary Condition	HPC	Harmonic Polynomial Cell
BF	Base Function	IBG	Immersed Boundary Grid
BFOG	Boundary-Fitted Overlapping Grid	IBOG	Immersed Boundary Overlapping Grid
BEM	Boundary Element Method	LE	Local Expression
BVP	Boundary Value Problem	LP	Lagrange Polynomial
DD	Domain Decomposition	LPC	Lagrange Polynomial Cell
FDM	Finite Difference Method	RK4	4^{th}-order Runge-Kutta
FVM	Finite Volume Method		

1. Introduction

The Harmonic Polynomial Cell (HPC) method is a field method initially proposed by Shao and Faltinsen [1-3] to solve the Laplace equation in terms of an unknown velocity potential. The Laplace equation is important in several fields, e.g. fluid dynamics, thermodynamics, electromagnetism and astronomy. In the HPC method, the local expression (LE) of the velocity potential within a cell uses harmonic polynomials (HPs) instead of traditional Lagrange polynomials (LPs). Hence, the

[*] Correspondence to: Shaojun Ma, AMOS, Department of Marine Technology, NTNU, Otto Nielsens veg 10, 7491Trondheim, Norway.
[†] E-mail: shaojun.ma@ntnu.no

governing equation is satisfied naturally. The connectivity between different cells is built by overlapping the LEs. This enables us to solve a Laplace problem throughout a domain with certain conditions specified along its boundary.

A key feature of the HPC method is in using higher-order local expressions satisfying Laplace equation, which means that we can expect a better accuracy than for many other field and boundary-integral formulations presently used. Moreover, the HPC method operates with a sparse coefficient matrix, so that many existing numerical matrix solvers can solve the associated problem efficiently. These advantages have already been demonstrated in [1, 3]. Shao and Faltinsen [1] compare the HPC method with a Lagrange Polynomial Cell (LPC) method. Both these methods are field solvers, and the difference lays in either using HPs or LPs for the LE. The results in [1] show that the CPU-time consumed by these two field solvers is similar, but the HPC method gives smaller errors and faster grid-convergence. In [3], the HPC method is compared with a quadratic Boundary Element Method (BEM). BEMs are the most popular methods nowadays to solve the Laplace equation. A standard BEM distributes singularities with unknown strengths on the boundaries of the computational domain, which means the matrix that BEM operates with is smaller but denser when compared to HPC. Shao and Faltinsen [3] compare the errors at the boundaries from HPC and BEM. The HPC shows better accuracy, and for a given accuracy level, consumes much less CPU-time. These favorable properties in accuracy and efficiency indicate that the HPC method represents a promising solution strategy in simulations involving a large number of time steps and/or many different computation cases.

Recent studies have tried to modify the original HPC method in different contexts, showing that it has the potential to deal with various complicated problems. Liang et al. [4] coupled the HPC method with a local corner-flow solution based on a domain decomposition (DD) strategy to account for the singular-flow characteristics due to sharp corners. They also introduced a double-layer node technique to simulate a thin free shear layer shed from lifting bodies. Fredriksen et al. [5, 6] coupled a viscous Finite Volume Method (FVM) solver with the HPC method based on DD to simulate the wave-induced response of a floating body with a moonpool. The FVM was used to solve the Navier-Stokes equations in vicinity of the body, and the HPC method for potential flows was deployed in the rest of the domain. This type of DD implementation represents a compromise between accuracy and efficiency. Methods like FVM can solve equations that are more complicated. However, by only using them in the regions where viscous flow matters, significant reductions in the associated CPU-time cost are possible. In the remainder of the domain, where the Laplace equation applies, the more efficient HPC method could be used. Other studies that apply DD to couple different computational regions can be found in e.g. [7, 8]. Recently, Bardazzi et al. [9] generalized the original HPC method to solve the Poisson equation. In their paper, problems involving forcing terms with a strong singularity are solved successfully.

Considering the encouraging applications of the HPC method thus far, it is natural to expect both further developments and broader applications in the future. A rational implementation and interpretation of the HPC method requires an in-depth knowledge of the method itself. Actually, many fundamental questions have risen in the recent studies. They are connected, for example, to the possibility of 1) identifying/using a simple and explicit strategy in the selection of the HPs for the LE, of 2) estimating the theoretical accuracy of HPC through analytical expressions for the discretization error and its order of grid-size convergence. Moreover, complex and dynamic boundaries are commonly involved in problems of practical interest, and so far, it is not clear whether it is possible 3) to identify a best HPC strategy to deal with such cases. Finally, because the HPC method is relatively new, it is valuable 4) to assess strategies to enhance the accuracy of a certain HPC calculation without increasing the computational effort significantly.

Driven by these open research questions, a comprehensive study of the properties of HPC is carried out in this paper. All the discussions focus on applications in fluid mechanics, where the Laplace equation governs the potential flow of an incompressible fluid with velocity potential as the unknown. For convenience, the investigations only focus on the two-dimensional (2D) HPC method. Nevertheless, the findings can be regarded as a starting point to analyze 3D cases. The remainder of the paper is organized as follows: Section 2 analyzes the local properties of HPC. The discussions are firstly focused on how to choose HPs properly, and then on the characteristics of discretization errors in LE. Section 3 examines the global properties of HPC and focuses on the errors in the global solution (GS). Section 4 discusses boundary treatments in terms of the local and global properties of HPC. Different treatments for complex and dynamic boundaries are proposed. Comparison studies are carried out to show the advantages and disadvantages of different treatments for the boundary value problem of a circular cylinder in infinite fluid and of free-surface wave propagation. Then one of the discussed HPC formulations, implementing findings of this study, is verified against a fully-nonlinear BEM for a radiation problem in heave for a surface-piercing circular cylinder. In Section 5, the main conclusions are given with a general guidance for HPC implementations.

2. The HPC method: local properties

Here, the HPC solution strategy is introduced and its local properties are discussed in terms of functional representation of the velocity potential within a cell and related local discretization error. It includes the error convergence rate, its dependence on cell topology, its distribution within the cell and its behavior across cells with different dimensions. These aspects are important for assessing HPC accuracy in connection with distorted/stretched grids, overlapping grids and Quadtree grids.

2.1 Local expression based on harmonic polynomials

Before we explain how to represent a velocity potential in terms of HPs, we must define a star-shaped domain. In mathematics, a set S in the Euclidean space R^n is called a star-shaped domain if there exists a point $\boldsymbol{p}$ in S such that for any point $\boldsymbol{q}$ in S the line segment from $\boldsymbol{p}$ to $\boldsymbol{q}$ is in S. We limit ourselves to two dimensions and introduce a Cartesian coordinate system oxy with origin coinciding with $\boldsymbol{p}$. The HPs can be defined by considering the complex velocity potential $\varphi = (x + \mathrm{i}\, y)^n$, where $\mathrm{i} = \sqrt{-1}$ is the complex unit and n is either zero or a positive integer. Taking the real and imaginary part of φ for any n from 0 to ∞, we identify all the HPs $h_k(x, y)$ needed to construct a solution of the Laplace equation inside S as a linear combination of these HPs. Vekua [10] proved that the HPs represent a complete system of functions in any star-shaped domain. This means that the series converges in an integral sense. However, this does not guarantee convergence at points where flow singularities occur, such as sharp corners of a body. Furthermore, Laplace equation must hold anywhere in S. This is not true, for instance, if there is a thin free shear layer inside S with jumps in the velocity potential across the shear layer. Liang et al. [4] investigated relevant problems with sharp body corners and thin free shear layers.

In this paper, non-dimensional HPs $h'_k(x, y) = h_k(x/\bar{r}, y/\bar{r})$ are used and the characteristic length $\bar{r}$ is defined as proportional to the grid-size. Detailed formulations for $h_k(x, y)$ and $h'_k(x, y)$ are given in Table 2 of Appendix A. Hereafter, the non-dimensional HPs are always used and indicated as $h_k(x, y)$ for convenience.

In the HPC method, the computational domain is divided into cells that overlap with each other, and within each cell, the solution is governed by the Laplace equation. The region occupied by a cell is regarded as the star-shaped region S, and the exact (true) solution for the velocity potential $\varphi_{true}(x, y)$ in S can be expressed in terms of an infinite series expansion based on HPs

$$\varphi_{true}(x, y) = \sum_{k=1}^{\infty} a_{true,k} h_k(x, y), \tag{1}$$

where $a_{true,k}$ are corresponding coefficients. In the HPC method, N_{HP} HPs will be included to establish a local expression (LE) for the approximated velocity potential $\varphi_{calc}(x, y)$ as

$$\varphi_{calc}(x, y) = \sum_{k=1}^{N_{HP}} a_{calc,j(k)} h_{j(k)}(x, y), \tag{2}$$

where $h_{j(k)}(x, y)$ are the HPs chosen for LE and $a_{calc,j(k)}$ the corresponding coefficients, which are found in terms of the numerical solution (for the velocity potential) at N_{node} nodes. The index k represents the position number of HP in the approximated series (2) and $j(k)$ is the position number of HP in the complete series (1). Generally, the HPs used in (2) may not coincide with the N_{HP} lowest-order HPs, i.e. $j(k) \geq k$. This is because the following analysis highlights that it is not always appropriate to select the first N_{HP} HPs from $h_k(x, y)$, though it would be the natural choice.

Since we truncate the series representation to a finite number of HPs, this solution will generally include an error. This means that for a certain set of HPs $h_{j(k)}(x, y)$, $a_{calc,j(k)} \neq a_{true,j(k)}$ may happen. To ensure that the problem is solvable in the sense of a least-square method, the requirement $N_{HP} \leq N_{node}$ should always be satisfied. We can now represent the local problem for the velocity potential in matrix form, and solve for the unknown coefficients $a_{calc,j(k)}$. Let us define the matrix

$\mathbf{F}$ with components $F_{ik} = h_{j(k)}(x_i, y_i)$; here $i = 1 \sim N_{node}$ is the index of node, $k = 1 \sim N_{HP}$ is the position number of HP in the truncated series (2). We also define the vector $\boldsymbol{\varphi}_{calc}$ with components $\varphi_{calc,i} = \varphi_{calc}(x_i, y_i)$, and a vector $\boldsymbol{a}_{calc}$ whose components are the coefficients $a_{calc,j(k)}$. By applying (2) for all N_{node} nodes, an equation system $\boldsymbol{\varphi}_{calc} = \mathbf{F}\boldsymbol{a}_{calc}$ is obtained. The coefficients $\boldsymbol{a}_{calc}$ can be solved multiplying left and right sides of the equation system by $\mathbf{F}^T$ and inverting $\mathbf{F}^T\mathbf{F}$:

$$\boldsymbol{a}_{calc} = \left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T\boldsymbol{\varphi}_{calc} = \mathbf{B}\boldsymbol{\varphi}_{calc}, \tag{3}$$

$$\mathbf{B} = \left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T. \tag{4}$$

Generally, it is natural to choose $N_{HP} = N_{node}$ as done in the previous studies [1-5, 8], and in this case

$$\mathbf{B} = \mathbf{F}^{-1}. \tag{5}$$

By substituting (3) and defining a vector of HPs $\boldsymbol{f}$ with components $f_k = h_{j(k)}(x, y)$, (2) can be rewritten as

$$\varphi_{calc}(x, y) = \boldsymbol{f} \cdot \mathbf{B}\boldsymbol{\varphi}_{calc}. \tag{6}$$

This equation gives the solution at the generic point (x, y) as interpolation from the values of the solution calculated at N_{node} nodes. In the HPC method, the problem solution may need the calculation of either the velocity potential or its spatial derivative (velocity) in normal $\boldsymbol{n}$ direction at any discrete grid node, say (x_0, y_0). Let us indicate these two values, respectively, $\varphi_{calc,0} = \varphi_{calc}(x_0, y_0)$ and $\varphi_{n,calc,0} = \varphi_{n,calc}(x_0, y_0) = \partial\varphi_{calc}(x, y)/\partial n\big|_{(x_0, y_0)}$. The velocity potential is obtained from equation (6) as

$$\varphi_{calc,0} = \boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{\varphi}_{calc} \tag{7}$$

Here the vector $\boldsymbol{f}_0$ has components $f_{0,k} = h_{j(k)}(x_0, y_0)$. The velocity in $\boldsymbol{n}$ direction can be estimated by taking the gradient of (6), i.e. $\varphi_{n,calc}(x, y) = \boldsymbol{f}_n \cdot \mathbf{B}\boldsymbol{\varphi}_{calc}$ where $\boldsymbol{f}_n$ has components $\partial h_{j(k)}(x, y)/\partial n$. Moreover, the normal velocity at (x_0, y_0) is

$$\varphi_{n,calc,0} = \boldsymbol{f}_{n,0} \cdot \mathbf{B}\boldsymbol{\varphi}_{calc}. \tag{8}$$

Here $\boldsymbol{f}_{n,0}$ has components $\partial h_{j(k)}(x, y)/\partial n\big|_{(x_0, y_0)}$. In the following sections, the origin of the local coordinate system is for convenience set at the interpolated point, i.e. $(x_0, y_0) = (0, 0)$.

To estimated $\varphi_{calc,0}$ and $\varphi_{n,calc,0}$, one needs to invert matrix $\mathbf{F}$ (see (5)). The functions h_k are linearly independent, but the truncated series of $h_{j(k)}$ may lead to singularity of $\mathbf{F}$ depending on the node distribution in the cell and on the used local coordinate system. This implies constraints in the choice of $h_{j(k)}$ from the complete set of h_k. In [3] a general method to select HPs for the three-dimensional HPC method was introduced. It selects HPs from lower to higher orders, and repeatedly checks until a solvable equation system is achieved. This is a general method that can be implemented also in 2D. A more explicit and simpler strategy exists for the selection of HPs if the cell has a symmetric node distribution with respect to the interpolated point and in addition $N_{HP} = N_{node}$, as discussed in the following.

One of the coordinate axes must coincide with the cell's symmetry axis. Along this axis there will be $N_{sym} \geq 0$ nodes, while the remaining $N_{node} - N_{sym}$ nodes will be an even number due to symmetry. The HPs can then be divided into even and odd functions with respect to the symmetry axis. For example, if the *y* axis is defined along the symmetry axis, the even HPs have property $h_k(x, y) = h_k(-x, y)$ and the odd HPs have the property $h_k(x, y) = -h_k(-x, y)$. As detailed in Appendix A,

every HP is either an even or an odd function. Thus, the total number of HPs used in the LE consists of $N_{HP,even}$ even HPs and $N_{HP,odd}$ odd HPs. In order to avoid local singularities of the matrix $\mathbf{F}$, the following requirements must be satisfied:

$$\begin{cases} N_{HP,even} = \left(N_{node} + N_{sym}\right)/2 \\ N_{HP,odd} = \left(N_{node} - N_{sym}\right)/2 \end{cases} \tag{9}$$

as necessary but not sufficient condition. This rule is also applicable for interpolation methods using other types of base functions (BFs), as long as $N_{BF} = N_{node}$ and there exists a symmetric axis about which the BFs are either odd or even. Detailed proof for this is given in Appendix B. Node distributions with more than one symmetry axis and with two axes perpendicular to each other represent a special case. In this case, the *x* and *y* axes of the local coordinate system can be set to coincide with these two symmetry axes, respectively, and conditions (9) must be satisfied for both axes.

In the following, relationships (9) will be applied for several example cells as defined in Fig. 1. In the figure, the dashed lines indicate symmetry axes, green nodes are used to form the LE and red nodes mark the interpolation points. For the cell in Fig. 1a, $N_{node} = 8$ and there are two perpendicular symmetry axes. The *x* and *y* axes can thus be defined to coincide with these two symmetric axes, with conditions (9) applied separately for both axes. For the *x* axis, $N_{sym} = 2$, indicating that there should be 5 even and 3 odd HPs with respect to the *x* axis. Similarly for the *y* axis, $N_{sym} = 2$, and there should be 5 even and 3 odd HPs with respect to the *y* axis. From Table 2 in Appendix A, it is apparent that all these requirements will be satisfied if the first 8 HPs $\{h_j, j = 1 \sim 8\}$ are chosen. For the cell in Fig. 1b, $N_{node} = 8$ with two perpendicular symmetry axes, but without any node along any of them ($N_{sym} = 0$). Thus, with the *x* and *y* axes chosen in the same way as in Fig. 1a, there should be 4 even and 4 odd HPs with respect to both axes. In this case, $\{h_j, j = 1 \sim 7, 9\}$ should be used to form the LE, while using $\{h_j, j = 1 \sim 8\}$ leads to a singularity of $\mathbf{F}$. The discussion regarding Fig. 1a and Fig. 1b shows that using the first N_{HP} HPs in the LE, without any further consideration, may not always be a proper choice. The HPs in the LE should instead be decided based on the node distribution in the cell in respect to the interpolation point. For the cell in Fig. 1c, $N_{node} = 9$ and there is only a single symmetry axis with $N_{sym} = 3$. Either the *x* or *y* axis must be defined to coincide with the symmetry axis, and there should be 6 even and 3 odd HPs in respect of this axis. If we choose the *y* axis, as shown in Fig. 1c, $\{h_j, j = 1 \sim 8, 11\}$ should be used. If on the other hand we choose the *x* axis to coincide with the symmetry axis, the proper choice is $\{h_j, j = 1 \sim 8, 10\}$, while the previous choice $\{h_j, j = 1 \sim 8, 11\}$ leads to a singularity of $\mathbf{F}$. This indicates that the choice of HPs used in the LE should be consistent with the definition of the local coordinate system. The final example, as shown in Fig. 1d, has $N_{node} = 9$ and two perpendicular symmetry axes with $N_{sym} = 3$. Node 5 has the same coordinate as the interpolation point. The *x* and *y* axes are defined so that they coincide with the symmetry axes, and consequently, there should be 6 even and 3 odd HPs with respect to each axis. From Table 2 in Appendix A, $\{h_j, j = 1 \sim 4, 6 \sim 8, 10, 11\}$ will satisfy all the above requirements. However, these HPs will lead to a local singularity of $\mathbf{F}$, which occurs because relationships (9) are only a necessary but not sufficient condition.

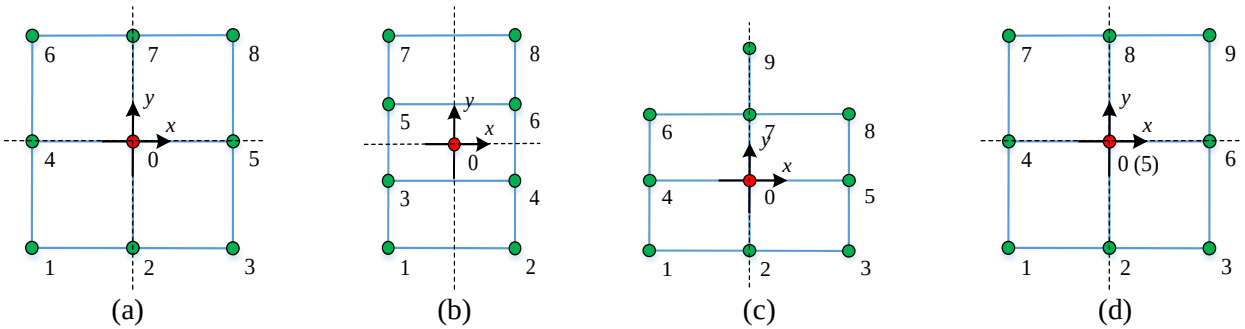


Fig. 1 Examples of cells with nodes distributed symmetrically

2.2 Discretization error of local expression

In this sub section, we derive the discretization error involved in the local expression (7) and discuss it for various implementations. The analysis will focus on the truncation error due to LE at the interpolated point $(0,0)$. In order to study the error solely due to the LE, we disregard the influence of possible inaccuracies in the solution at nodes (x_i, y_i) used in the interpolation. The latter relates to error propagation in a global problem and will be analyzed in Section 3. Thus, the solutions at the nodes (x_i, y_i) are assumed to coincide with the true values, i.e. $\varphi_{calc}(x_i, y_i) = \varphi_{true}(x_i, y_i)$. We collect these in a vector $\boldsymbol{\varphi}_{true}$ with elements $\varphi_{true,i} = \varphi_{true}(x_i, y_i)$, so that equation (7) becomes $\varphi_{calc,0} = \boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{\varphi}_{true}$. For convenience, a "best approximated" solution is defined as

$$\varphi_{best}(x,y) = \sum_{k=1}^{N_{HP}} a_{true,j(k)} h_{j(k)}(x,y). \tag{10}$$

At the interpolated point, $\varphi_{best,0} = \varphi_{best}(0,0) = \varphi_{true,0} = \varphi_{true}(0,0)$ and $\varphi_{n,best,0} = \partial\varphi_{best}(x,y)/\partial n\big|_{(0,0)} = \varphi_{n,true,0} = \partial\varphi_{true}(x,y)/n\big|_{(0,0)}$.

Similar derivations as for $\varphi_{calc}(x,y)$ can be made for $\varphi_{best}(x,y)$. We define a vector $\boldsymbol{\varphi}_{best}$ with elements $\varphi_{best,i} = \varphi_{best}(x_i, y_i)$, so that $\varphi_{best}(x,y) = \boldsymbol{f} \cdot \mathbf{B}\boldsymbol{\varphi}_{best}$. This gives

$$\varphi_{best,0} = \boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{\varphi}_{best} \tag{11}$$

and error in equation (7) can then be expressed as

$$e_{\varphi,0} = \varphi_{calc,0} - \varphi_{true,0} = \varphi_{calc,0} - \varphi_{best,0} = \boldsymbol{f}_0 \cdot \mathbf{B}\left(\boldsymbol{\varphi}_{true} - \boldsymbol{\varphi}_{best}\right). \tag{12}$$

We define $h_{l(k)}(x,y)$ as the HPs that have not been chosen for the representation (2) of the solution, whose exact coefficients in (1) are written as $a_{true,l(k)}$. Then, according to equations (1) and (10), $\varphi_{true}(x,y) - \varphi_{best}(x,y) = \sum_{k=N_{HP}+1}^{\infty} a_{true,l(k)} h_{l(k)}(x,y)$. Thus if we define a vector $\boldsymbol{g}_k$ with elements $g_{k,i} = h_{l(k)}(x_i, y_i)$, $e_{\varphi,0}$ becomes

$$e_{\varphi,0} = \sum_{k=N_{HP}+1}^{\infty} a_{true,l(k)} \left(\boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{g}_k\right). \tag{13}$$

The coefficients $a_{true,l(k)}$ depend only on the local true solution, and $\boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{g}_k$ depends only on the node distribution. A similar formulation applies for the error in the velocity in $\boldsymbol{n}$ direction, i.e.

$$e_{\partial\varphi/\partial n,0} = \sum_{k=N_{HP}+1}^{\infty} a_{true,l(k)} \left(\boldsymbol{f}_{n,0} \cdot \mathbf{B}\boldsymbol{g}_k\right). \tag{14}$$

2.2.1 Local grid convergence

A key feature of the discretization error is its dependence on the grid refinement. This is measured by the grid convergence order p, and similar to [11], it is defined here by assuming that the discretization error can be expressed as $e = \alpha_p \bar{r}^p + o\left(\bar{r}^p\right)$ with $\alpha_p \neq 0$. The convergence order can then be determined directly from expressions (13) and (14), and depends on N_{HP} and on the distribution of nodes. We define p_φ and $p_{\partial\varphi/\partial n}$ as the convergence orders of velocity potential and velocity in $\boldsymbol{n}$ direction, respectively. It is generally found that $p_\varphi = M$ and $p_{\partial\varphi/\partial n} = M - 1$ with M the lowest order of HP not used for the approximation (2) of the solution, i.e. the lowest order of HP in $h_{l(k)}(x,y)$. The error in velocity convergences one order slower than that of the velocity potential. This is simply due to the fact that $\boldsymbol{f}_0\mathbf{B} \sim O(1)$ and $\boldsymbol{f}_{n,0}\mathbf{B} \sim O\left(\bar{r}^{-1}\right)$.

An interesting issue related to implementation is whether we can improve the grid convergence by using a particular strategy for the node distribution. We find this to be possible, especially for grids with symmetry features. Assuming $N_{HP} = N_{node} = 8$ and using $\{h_j, j = 1 \sim 8\}$ for the LE, the grid convergence for different types of node distributions is summarized in Table 1. $\boldsymbol{n}_{sym}$ is a unit vector along a symmetry axis, while $\boldsymbol{n}$ is an arbitrary unit vector that may not coincide with any symmetry axis. In this case $M = 4$, and for an arbitrary node distribution, the convergence orders are $p_\phi = M = 4$ and $p_{\partial\varphi/\partial n} = M - 1 = 3$. The convergence order for ϕ can be significantly improved by symmetry, and p_φ becomes 8 when the grid is squared (a squared cell automatically has two symmetry axes). The error in velocity can also be improved by proper choice of the local node distribution, although not as much as for the velocity potential.

Table 1. Order of grid convergence for the error related to the local expression ($N_{HP} = N_{node} = 8$)

Node distribution	Error in φ	Error in $\partial\varphi/\partial n_{sym}$	Error in $\partial\varphi/\partial n$
Arbitrary	4	-	3
With one symmetry axis	5	4	3
With two symmetry axes	6	4	4
Squared grid	8	4	4

2.2.2 Influence of cell topology on the local error

In this section, we analyze the effect of spatial variations in the local grid geometry on the numerical error. Two situations of practical importance are grid stretching and grid distortion. Grid stretching means that the grid is refined in one direction while keeping the grid size unchanged in the other. This is often seen in connection with Computational Fluid Dynamic (CFD) methods [12, 13], with the aim to improve accuracy near a domain boundary by refining the grid in the direction of strong flow gradients. An example of such scenario is shown in Fig. 2. Grid distortion means that the cell edges are not perpendicular to each other and can occur for boundary-fitted grids. An example is shown in Fig. 3.

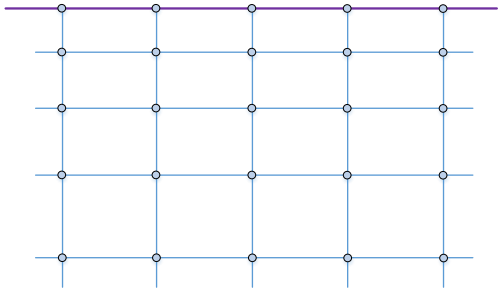


Fig. 2 Refinement in one direction (stretching)

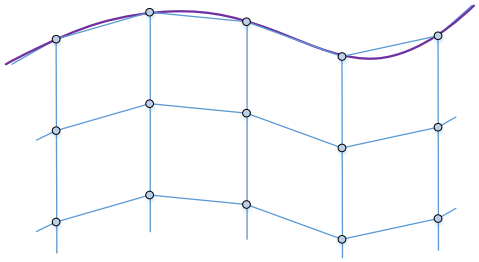


Fig. 3 Distortion due to boundary geometry

There are many factors that influence the accuracy, including the convergence rate, grid size and features of the true solution when the grid is stretched or distorted. As shown in Section 2.2.1, many low order error terms in (13) and (14) become zero for a squared cell. This suggests as a hypothesis that it may be possible to achieve a smaller error with squared cells in a coarse grid than with non-squared cells in a more refined grid in many cases. Without loss of generality, we consider the following true solution as an example:

$$\varphi_{true}(x,y) = \sum_{k=1}^{17} a_{true,k} h_k(x,y), \qquad a_{true,k} = 1 \ \text{for}\ k = 1 \sim 17. \tag{15}$$

The calculation is carried out for a single cell, initially chosen to be squared with $dx = dy = 0.1$. A contour plot of the true solution specified by equation (15) for such cell is shown in Fig. 4. We use $N_{HP} = N_{node} = 8$ to form the LE and examine the effect of grid stretching and distortion through two simplified cases. As shown in Fig. 5a and Fig. 6a, the Dirichlet B.C.s are specified in the cell's edge nodes #1~#8, and the error is computed in the interior node (node #0). For sake of comparison, the calculations have also been carried out using a Finite Difference Method (FDM) from [1]. This FDM uses LPs instead

of HPs, and becomes the five-point stencil FDM, commonly used e.g. for marine applications, when the cell has edges perpendicular to each other.

(1) Stretched cell

This case corresponds to the principal scenario in Fig. 2. As shown in Fig. 5a, the initial squared cell is stretched into a rectangular cell by changing dx while keeping fixed $dy = 0.1$ and maintaining unchanged the position of node #0. This corresponds to the principal scenario in Fig. 2. The aspect ratio defined as dx/dy is used to measure how much the grid has been stretched. The errors in velocity potential and velocity components are plotted in Fig. 5b as a function of the cell aspect ratio. For the FDM, the error in velocity potential decreases monotonically without any local minimum as the aspect ratio decrease and therefore as the cell dimension is reduced. Thus, grid refinement in one direction, i.e. the stretching, will in general improve the accuracy in the FDM. The same is not true for the HPC method. In this case, a local minimum is observed for the error in velocity potential when the cell is nearly squared. Then, for increasing refinement in *x*-direction, it increases towards a local maximum before it tends again to reduce. Similar behavior occurs for the velocities, although the local-minimum regions deviate slightly from a squared cell. This suggests that the safest choice for the HPC method is to keep the cell's aspect ratio close to unity and avoid grid stretching in one direction. Section 4 provides useful information for performing local refinement, if needed, in the HPC method.

(2) Distorted cell

As shown in Fig. 6a, the initial squared cell is distorted into a parallelogram by changing the angle γ while maintaining a fixed aspect ratio $dx = dy = 0.1$. This represents an idealized modelling for the distortion illustrated in Fig. 3. The skewness of the cell, defined as $\cot\gamma$, is used as a measure of the grid distortion. The grid is squared when $\cot\gamma = 0$, while its upper part is distorted towards right when $\cot\gamma > 0$, and towards left when $\cot\gamma < 0$. Errors in the velocity potential and velocity components as a function of the cell skewness are shown in Fig. 6b. For the FDM , the error of velocity potential can have some local minimal due to the true solution behavior, and the minimal error may not happen when the skewness is close to 0. For the HPC , the numerical error of the velocity potential is minimum when the cell is nearly squared, and increases dramatically when the cell is distorted. However, the minimum error is not found exactly at $\cot\gamma = 0$. The error of the velocity components has a more complicated variation with respect to cell skewness. This is due to the behavior of the true solution used in this particular study. On the basis of these results, for the HPC method the safest option from a general point of view is to apply a cell with zero or very limited skewness.

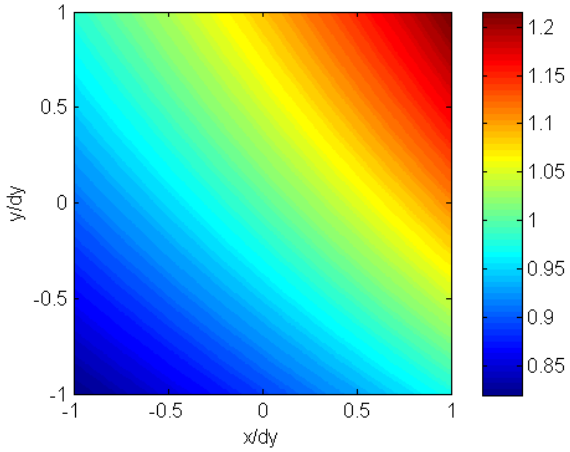
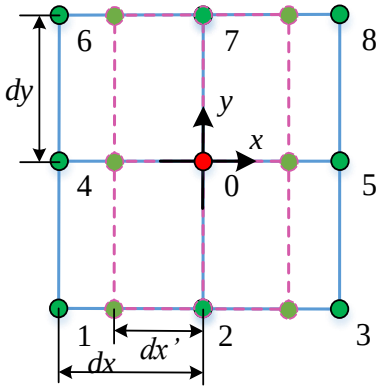
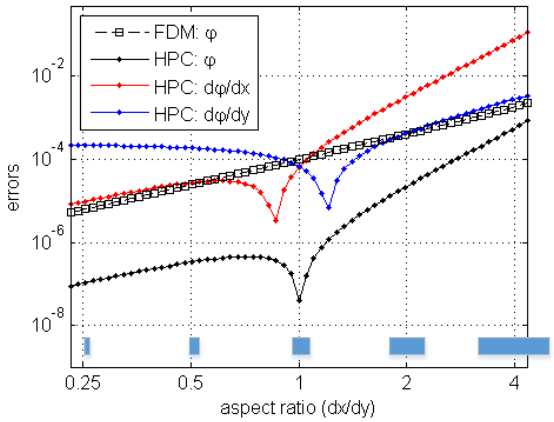


Fig. 4 Contour plot of φ_{true}

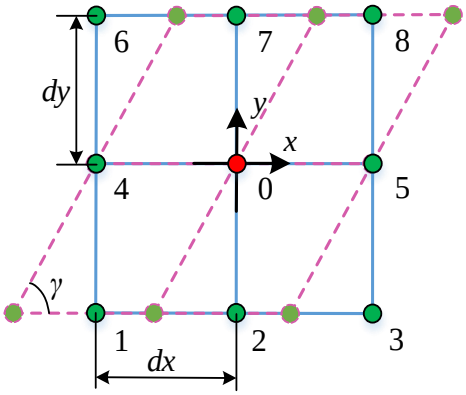


(a) stretched cell

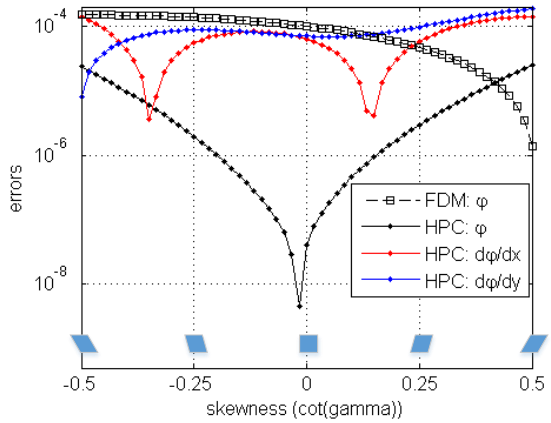


(b) errors

Fig. 5 Error as a function of cell aspect ratio (dx/dy)



(a) distorted cell



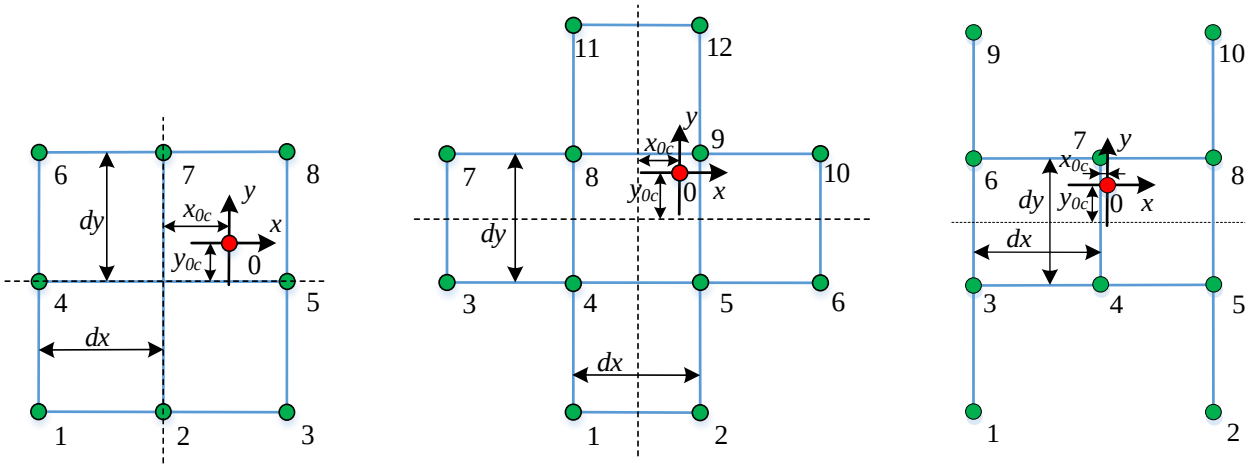
(b) errors

Fig. 6 Error as a function of cell skewness ($\cot\gamma$)

2.2.3 Local error distribution within a cell

For a predefined cell, it is interesting to determine how the error will change for points at different locations within the cell. This is particularly interesting with respect to overlapping and interpolation between different grids and for immersed boundary methods. Such methods are discussed in Section 4. From Table 1, a better accuracy is generally expected for points located close to the cell center, since in this case the surrounding node distribution will be more symmetric.

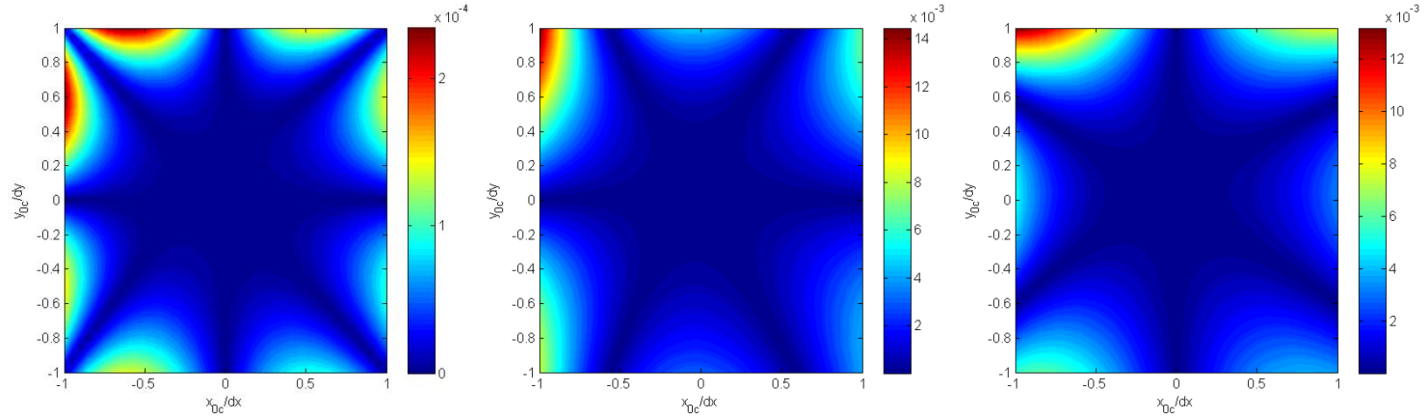
As an example to confirm this, the same problem as examined in Section 2.2.2 and with true solution (15) is studied for a squared grid with $dx = dy = 0.1$. For the commonly used cell with $N_{HP} = N_{node} = 8$ shown in Fig. 7a, (x_{0c}, y_{0c}) is defined as the location of the interpolated point relative to the cell center. The errors in potential and velocity components for different locations within the cell are contoured in Fig. 8. The error distributions are not exactly symmetric in respect of the symmetry axes of the cells, which is because the true solution is asymmetric. Nevertheless, the errors are smaller at locations closer to the cell center, and they increase dramatically when approaching the cell edges. This suggests that it is profitable from an accuracy point of view to keep the interpolation points within the central region of a cell. Moreover, several axes exist within the cell where the errors remain small even towards the cell edges. These axes are related to the distribution of nodes and are different for potential and velocity components. Because of this, it is difficult to utilize these features for general cases.



(a) $N_{node}=8$ (b) $N_{node}=12$ (c) $N_{node}=10$

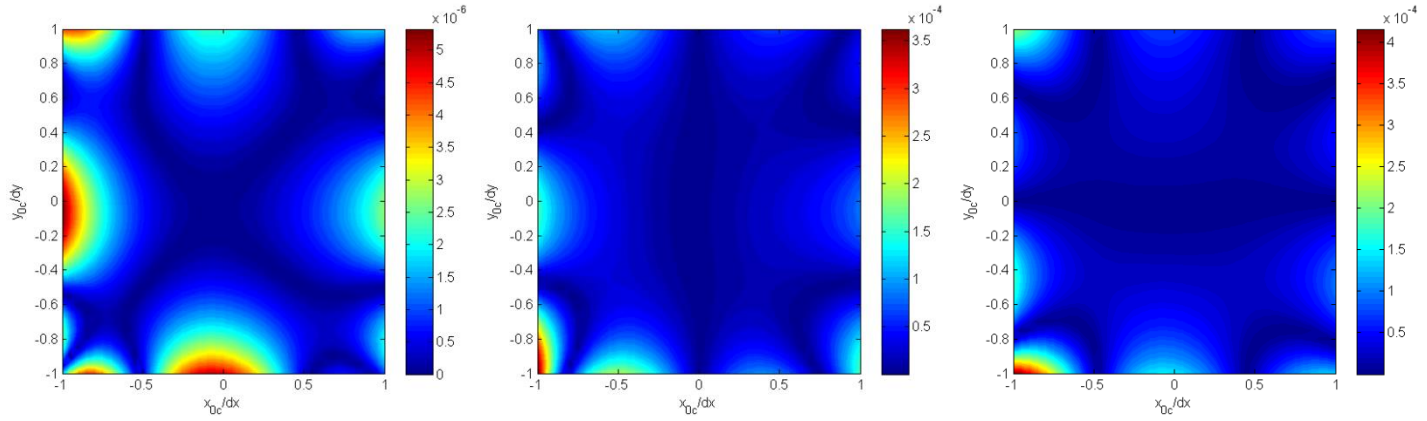
Fig. 7 Cells for study on error distribution

Occasionally it may happen that an interpolation point is not close to the cell center, for example if $(x_{0c}, y_{0c}) \approx 0.5(\pm dx, \pm dy)$. In this case, a loss of accuracy seems to be unavoidable if we use the cell definition in Fig. 7a. One solution to overcome this problem is to use higher order cells. An example of such a cell is shown in Fig. 7b. Here $N_{HP}=12$, with the $N_{node}=12$ closest grid nodes, are used to form the LE. To avoid possible local singularity as discussed in Section 2.1, the first 6 even HPs and the first 6 odd HPs are chosen, which correspond to $\{h_j, j=1\sim 11,13\}$ from Table 2 in Appendix A. Here h_{12} is omitted since it may cause a singularity when $x_{0c}\approx 0$. The error distributions for the higher-order cell are shown in Fig. 9. The error contours have similar pattern as for the $N_{HP}=N_{node}=8$ case, but the absolute values of the error are some orders of magnitude smaller. This indicates that the accuracy is improved with the higher-order cell, and also that we can achieve more flexibility in placing the interpolation points. This type of cell with $N_{HP}=N_{node}=12$ works for general scenarios. If the interpolation point is close to a grid line (a line connecting two neighbor nodes), a simpler $N_{HP}=N_{node}=10$ cell can be applied. Fig. 7c shows such a 10-node cell where the interpolation point is located close to the grid line between nodes 4 and 7, i.e. $x_{0c}\approx 0$ and $|y_{0c}|\le 0.5dy$. In order to avoid a local singularity, the first 6 even HPs and the first 4 odd HPs are used, corresponding to $\{h_j, j=1\sim 9,11\}$. The errors are contoured in Fig. 10, showing similar behavior as for the $N_{HP}=N_{node}=8$ cell, but the absolute values are smaller especially for the error in φ and in $\partial\varphi/\partial y$.



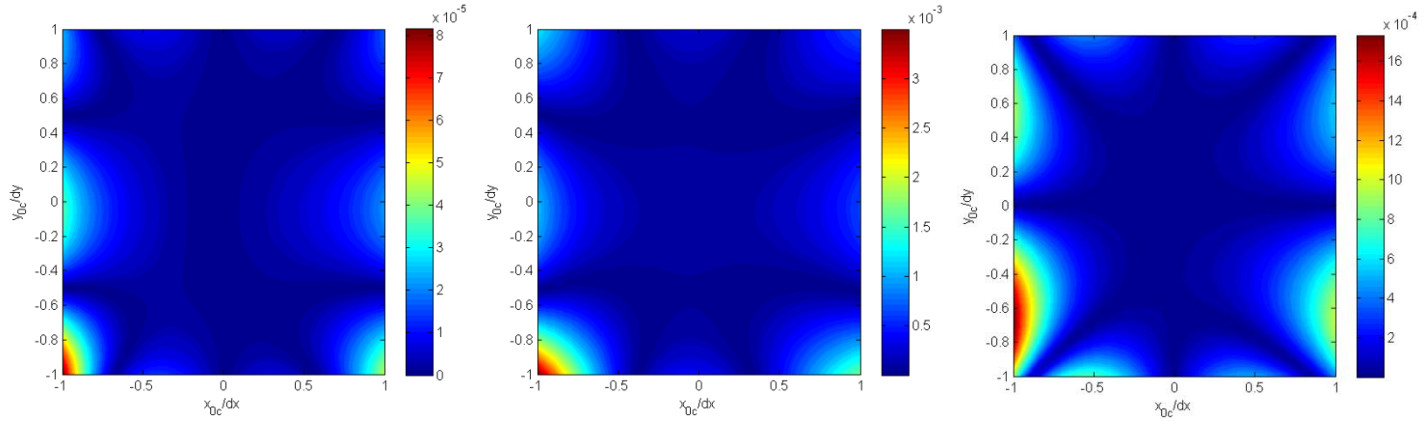
(a) error in φ (b) error in $\partial\varphi/\partial x$ (c) error in $\partial\varphi/\partial y$

Fig. 8 Error distribution in a cell with $N_{HP}=N_{node}=8$



(a) error in φ (b) error in $\partial\varphi/\partial x$ (c) error in $\partial\varphi/\partial y$

Fig. 9 Error distribution in a cell with $N_{HP} = N_{node} = 12$



(a) error in φ (b) error in $\partial\varphi/\partial x$ (c) error in $\partial\varphi/\partial y$

Fig. 10 Error distribution in a cell with $N_{HP} = N_{node} = 10$

2.2.4 Local error across cells with different dimensions

From the above discussions as well as verified in practice, it is generally found advantageous to use squared grids. Moreover, in practical applications one may tend to refine the grid locally in order to achieve a better overall accuracy without increasing the total number of cells unnecessarily. If the squared-grid feature is to be maintained during this refinement, grid systems with different cell sizes have to be used in the domain. These grids can communicate either through overlapping or Quadtree methods. Section 2.2.3 discussed some aspects of the LE relevant in the case of overlapping grids. Here the discussion will focus on the LE for Quadtree grids. Fig. 11 shows an example of a typical Quadtree grid for a 2D HPC method with one level of sub-division, i.e. there is only a child grid with cell area equal to $1/4$ of the parent grid. The purple and yellow points are the nodes from the parent and child grids, respectively, and their LEs emerge directly from the grid system to which they belong. The blue points are nodes that lay on the border of the refined-level grid. The LEs for these nodes must be constructed so to ensure a proper communication between the two grid systems.

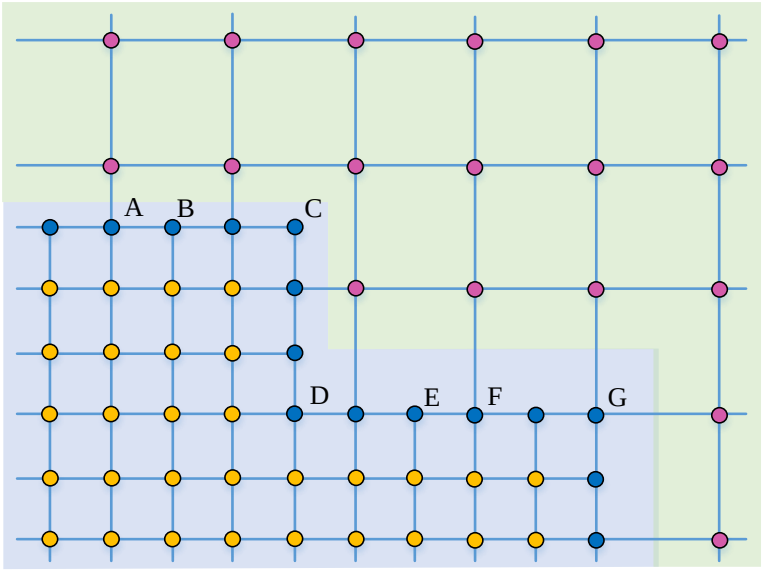
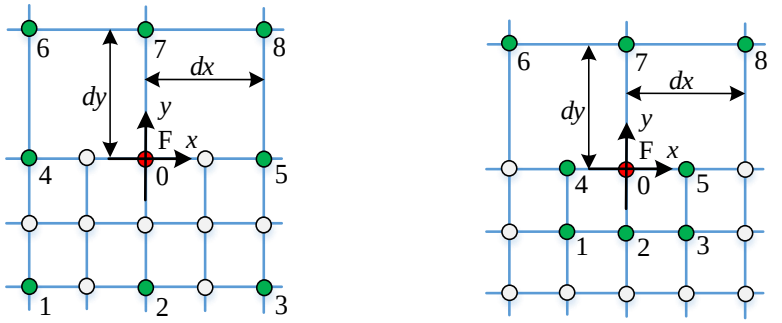


Fig. 11 Typical Quadtree grid

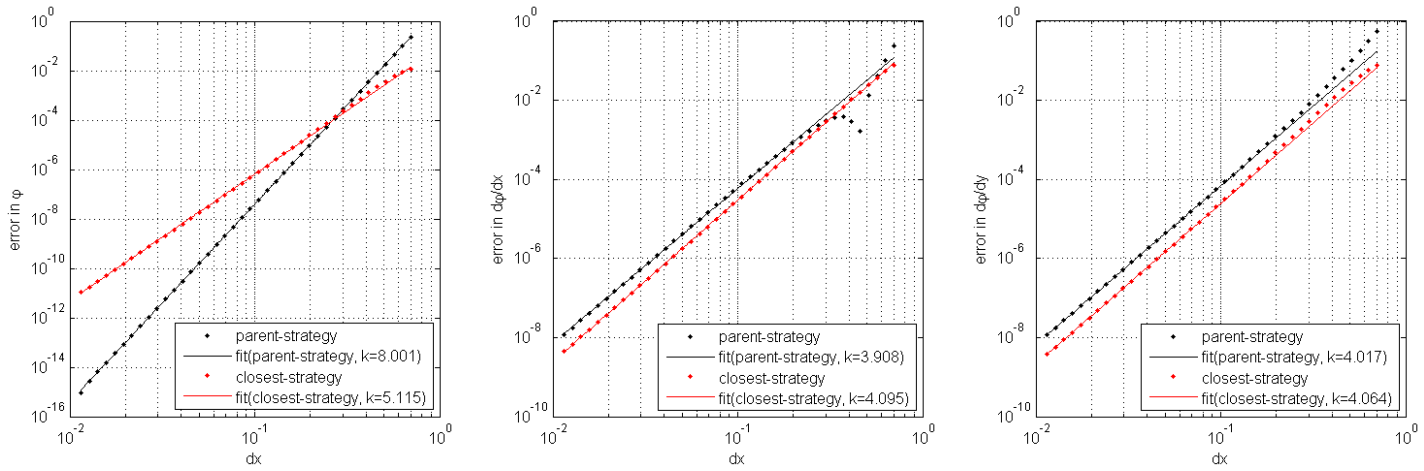
A fundamental question in this method is which nodes should be used to form the LE for the blue points (border nodes) in the child grid of Fig. 11. Two typical choices exist: One is to use only the nodes in the parent grid, which we expect may lead to a better grid convergence since the cells are more symmetric. Another choice is to use the closest nodes irrespective of which grid they belong to, so that the average grid size and possibly the error are minimized. We hereafter refer to the two strategies as "parent" and "closest". Generally, the "closest" method may perform better for coarse grids, while the "parent" method can have an advantage for fine grids. Taking node F in Fig. 11 as an example, Fig. 12 shows the nodes used in the LE for the two strategies. The green nodes are those used for interpolation while the white nodes are inactive.

We solve the one-cell problem defined in Section 2.2.2 with true solution (15). The Quadtree-grid method is used with the two aforementioned interpolation strategies and varying the grid size systematically. As shown in Fig. 12, Dirichlet B.C. is given to the nodes #1~#8, and the error is calculated at the center node #0. The grids are kept squared during refinement, and the parent grid size $dx = dy$ is used to evaluate the convergence rate. Fig. 13 shows the errors in velocity potential and velocity components as a function of grid size. For the velocity potential, the "parent" strategy gives an 8^{th}-order convergence and has smaller error than the "closest" strategy when the grid-size is small. The "closest" strategy results in a 5^{th}-order convergence for the velocity potential, which relates to the fact that there is only one symmetry axis in this case (see Fig. 12b). However, the absolute value of the error is smaller than for the "parent" strategy for very coarse grids. With respect to velocity components, both strategies lead to nearly 4^{th}-order convergence. The "closest" strategy appears to perform slightly better, especially for the coarsest grids. The seemingly strange behavior of the error in $\partial\varphi/\partial x$ for large grid sizes is due to the features of the true solution. In practice, within a Quadtree-grid method, it is the velocity potential, not the velocity components, that is directly involved in the communication between the grids. More discussions about this will be given in Section 3. Furthermore, we usually avoid very large grid sizes in order to capture all the important flow features involved. Thus, the "parent" interpolation strategy may be a more suitable choice for border nodes.



(a) Using only "parent" nodes (b) Using "closest" nodes

Fig. 12 Different choices of nodes for local expression in Quadtree grid



(a) error in φ (b) error in $\partial\varphi/\partial x$ (c) error in $\partial\varphi/\partial y$

Fig. 13 Grid error convergence for different choices of nodes in Quadtree grid

In the above example, the border node F coincides with a node in the parent grid system, but there also exist many other border nodes not coinciding with any node from the parent grid system. In this case, different forms of LEs can be implemented. This is exemplified in Fig. 11: Nodes A, F, G can use the interpolation cell in Fig. 12a, nodes D and E can use an interpolation cell similar to Fig. 7c with $x_{0c} = y_{0c} = 0$, and nodes B and C can use an interpolation cell similar to Fig. 7b with $x_{0c} = y_{0c} = 0$.

3. The HPC method: global properties

Here, the global properties of the method are analyzed in terms of global error and its convergence rate. The contributions to the global error come from inside the fluid domain and from the boundary conditions.

3.1 Global error: contributions and theoretical convergence rate

In the HPC method [1-3], the global matrix system is built by imposing the local equation (7) at every node inside the computational domain. The boundary condition is set directly for nodes along Dirichlet boundaries. For nodes along Neumann boundaries, the boundary condition is set in the form of equation (8). Let us consider a problem with N_{total} nodes involved in the computational domain. The resulting global matrix system is a set of linear equations in the form

$$\mathbf{A}\boldsymbol{\varphi}_{calc,GS} = \boldsymbol{b}_{\phi}, \qquad (16)$$

where $\mathbf{A}$ is a $N_{total} \times N_{total}$ coefficient matrix, $\boldsymbol{b}_{\phi}$ is a $N_{total} \times 1$ boundary condition (B.C.) vector, and $\boldsymbol{\varphi}_{calc,GS}$ is the global solution (GS) to be determined at every node throughout the global domain.

In Section 2.2, the discussion for the error at interpolation points assumes the solution at surrounding nodes to coincide with the true solution. This was made to study the convergence properties of the solution locally, but it is not true when we consider the problem in a global sense. Considering this, we have to modify the local error expression for the GS. For the same cell as in Section 2.2, we substitute $\varphi^{GS}_{true}(x,y)$ and $\varphi^{GS}_{best}(x,y)$ to $\varphi_{true}(x,y)$ and $\varphi_{best}(x,y)$, respectively. At the origin of the cell coordinate system, $\varphi^{GS}_{best,0} = \varphi^{GS}_{best}(0,0) = \varphi^{GS}_{true,0} = \varphi^{GS}_{true}(0,0)$, $\varphi^{GS}_{n,best,0} = \partial\varphi^{GS}_{best}(x,y)/\partial n\big|_{(0,0)} = \varphi^{GS}_{n,true,0} = \partial\varphi^{GS}_{true}(x,y)/\partial n\big|_{(0,0)}$. Moreover, $\varphi^{GS}_{calc}(x,y)$ is the calculated global solution in the cell. This generally does not coincide with $\varphi^{GS}_{true}(x,y)$ at nodes (x_i, y_i) at the cell boundary. We define the vectors $\boldsymbol{\varphi}^{GS}_{calc}$, $\boldsymbol{\varphi}^{GS}_{true}$ and $\boldsymbol{\varphi}^{GS}_{best}$ in the same way as in Section 2.2. Thus, expression (12) becomes

$$
\begin{aligned}
e_{\varphi,0}^{GS} &= \varphi_{calc,0}^{GS} - \varphi_{true,0}^{GS} = \varphi_{calc,0}^{GS} - \varphi_{best,0}^{GS} = \boldsymbol{f}_0 \cdot \mathbf{B}\left(\boldsymbol{\varphi}_{calc}^{GS} - \boldsymbol{\varphi}_{best}^{GS}\right) \\
&= \boldsymbol{f}_0 \cdot \mathbf{B}\left(\boldsymbol{\varphi}_{true}^{GS} - \boldsymbol{\varphi}_{best}^{GS}\right) + \boldsymbol{f}_0 \cdot \mathbf{B}\left(\boldsymbol{\varphi}_{calc}^{GS} - \boldsymbol{\varphi}_{true}^{GS}\right) \\
&= \left(\sum_{k=N_{HP}+1}^{\infty} a_{true,l(k)} \left(\boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{g}_k \right) \right) + \boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{e}_{\varphi}^{GS}
\end{aligned} \quad , \tag{17}
$$

where $\boldsymbol{e}_{\varphi}^{GS} = \boldsymbol{\varphi}_{calc}^{GS} - \boldsymbol{\varphi}_{true}^{GS}$ is the error vector for the velocity potential at nodes (x_i, y_i). The last line of (17) shows that the global error $e_{\varphi,0}^{GS}$ consists of two parts. The first part is the same as the local error $e_{\varphi,0}$ in expression (13), which reflects the error due to truncation of the HP-based LE. The second part reflects inaccuracy of the solution at nodes (x_i, y_i) used for the LE, which can be interpreted as propagation error. The error in velocity at a Neumann B.C. is expressed in a similar way:

$$
e_{\partial\varphi/\partial\mathbf{n},0}^{GS} = \left(\sum_{k=N_{HP}+1}^{\infty} a_{true,l(k)} \left(\boldsymbol{f}_{n,0} \cdot \mathbf{B}\boldsymbol{g}_k \right) \right) + \boldsymbol{f}_{n,0} \cdot \mathbf{B}\boldsymbol{e}_{\varphi}^{GS} \quad . \tag{18}
$$

Let us define a $N_{total} \times 1$ vector $\boldsymbol{e}_{\varphi,GS}$ that contains the error in the velocity potential at every node in the global domain, i.e. $\boldsymbol{e}_{\varphi,GS} = \boldsymbol{\varphi}_{calc,GS} - \boldsymbol{\varphi}_{true,GS}$ with $\boldsymbol{\varphi}_{true,GS}$ defined as the exact solution at every node throughout the domain. By considering $\boldsymbol{e}_{\varphi,GS}$ as our unknown, a numerical solution for this global-error vector is directly achievable without first computing the velocity potential. In particular, we note that expressions (17) and (18) share the same coefficients as expressions (7) and (8), which indicates that we can estimate the global error solving the equation system:

$$
\mathbf{A}\boldsymbol{e}_{\varphi,GS} = \boldsymbol{b}_e \quad , \tag{19}
$$

where $\mathbf{A}$ is the same coefficient matrix as in equation system (16) for the velocity potential. The right-hand side $\boldsymbol{b}_e$ is a $N_{node} \times 1$ disturbance vector consisting of three parts:

$$
\begin{aligned}
&\boldsymbol{b}_e = \boldsymbol{b}_e^I + \boldsymbol{b}_e^D + \boldsymbol{b}_e^N \\
&\begin{cases}
\boldsymbol{b}_e^I = \displaystyle\sum_{k=N_{HP}+1}^{\infty} a_{true,l_I(k)} \left(\boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{g}_k \right) \\
\boldsymbol{b}_e^D = \displaystyle\sum_{k=N_{HP}+1}^{\infty} a_{true,l_D(k)} \left(\boldsymbol{f}_0 \cdot \mathbf{B}\boldsymbol{g}_k \right) \\
\boldsymbol{b}_e^N = \displaystyle\sum_{k=N_{HP}+1}^{\infty} a_{true,l_N(k)} \left(\boldsymbol{f}_{n,0} \cdot \mathbf{B}\boldsymbol{g}_k \right)
\end{cases} .
\end{aligned} \tag{20}
$$

$\boldsymbol{b}_e^I$ is the contribution from discretization errors inside the domain, $\boldsymbol{b}_e^D$ is associated with Dirichlet B.C.s, and $\boldsymbol{b}_e^N$ is associated with Neumann B.C.s. Since (19) is a linear system of equations, the error can also be divided into three corresponding parts $\boldsymbol{e}_{\varphi,GS} = \boldsymbol{e}_{\varphi,GS}^I + \boldsymbol{e}_{\varphi,GS}^D + \boldsymbol{e}_{\varphi,GS}^N$ that can be solved separately

$$
\mathbf{A}\boldsymbol{e}_{\varphi,GS}^K = \boldsymbol{b}_e^K, \quad K = I, D, N \quad . \tag{21}
$$

$\boldsymbol{e}_{\varphi,GS}^I$ can be understood as the error induced by discretization of the governing equation. Its corresponding equation (21) has error contributions from everywhere inside the computational domain. Thus the upper bound of $\boldsymbol{e}_{\varphi,GS}^I$ can be estimated by the condition number of the global matrix

$$
E_{\varphi,GS}^I = \frac{\left\| \boldsymbol{e}_{\varphi,GS}^I \right\|_2}{\left\| \boldsymbol{\varphi}_{true,GS} \right\|_2} \approx \frac{\left\| \boldsymbol{e}_{\varphi,GS}^I \right\|_2}{\left\| \boldsymbol{\varphi}_{calc,GS} \right\|_2} \le \mathrm{cond}(\mathbf{A}) \cdot \frac{\left\| \boldsymbol{b}_e^I \right\|_2}{\left\| \boldsymbol{b}_{\varphi} \right\|_2} , \tag{22}
$$

where $\|\ \|_2$ is the L_2 norm, and for a matrix $\mathbf{X}$ with dimensions $M_X \times N_X$, it is defined as $\|\mathbf{X}\|_2 = \sqrt{\sum_{i=1}^{M_X}\sum_{j=1}^{N_X}\left|X_{ij}\right|^2}$. More explanation about (22) can be found in Appendix C. Expression (22) shows that the grid convergence of the L_2 error $E^I_{\phi,GS}$ depends on both $\mathrm{cond}(\mathbf{A})$ and $\|\boldsymbol{b}^I_e\|_2 / \|\boldsymbol{b}_\varphi\|_2$. The latter term can be expressed as $\|\boldsymbol{b}^I_e\|_2 / \|\boldsymbol{b}_\varphi\|_2 = \alpha_{p^I_{LE}} \bar{r}^{\,p^I_{LE}} + o\left(\bar{r}^{\,p^I_{LE}}\right)$, where p^I_{LE} is the local convergence rate for cells inside the domain as discussed in Section 2.2.1. Similarly, we can express the condition number as $\mathrm{cond}(\mathbf{A}) = \alpha_{p_{cond}} \left(1/\bar{r}\right)^{p_{cond}} + o\left(\left(1/\bar{r}\right)^{p_{cond}}\right)$. This means that we can approximate the global convergence rate of $E^I_{\varphi,GS}$ during grid refinement as $p^I_{GS} \approx p^I_{LE} - p_{cond}$. Experience has shown that we can achieve a more proper definition of the condition number of the coefficient matrix $\mathbf{A}$ by eliminating the rows related with boundary conditions. The reason behind this observation still needs to be confirmed theoretically, which can be a scope for future work. Generally $p_{cond} > 0$, so that the global convergence is slower than the local convergence.

The errors $\boldsymbol{e}^D_{\varphi,GS}$ and $\boldsymbol{e}^N_{\varphi,GS}$ are induced by the Dirichlet and Neumann B.C.s respectively, and the estimation for them is equivalent to solving a linear BVP. Taking $\boldsymbol{e}^N_{\varphi,GS}$ as an example, it is governed by the Laplace equation in the same way as the velocity potential, and gives an equation system (21) with $K = N$. Here $\boldsymbol{b}^N_e$ gives the B.C.s for the BVP. In this case, $\|\boldsymbol{e}^N_{\varphi,GS}\|_2$ is proportional to $\|\boldsymbol{b}^N_e\|_2$. Following a similar analysis as for $\boldsymbol{e}^I_{\phi,GS}$, $\|\boldsymbol{b}^N_e\|_2 / \|\boldsymbol{b}_\varphi\|_2 = \alpha_{p^N_{LE}} \bar{r}^{\,p^N_{LE}} + o\left(\bar{r}^{\,p^N_{LE}}\right)$ where p^N_{LE} is the local convergence order of the velocity at Neumann boundaries. The convergence order of $E^N_{\varphi,GS} = \|\boldsymbol{e}^N_{\varphi,GS}\|_2 / \|\boldsymbol{\varphi}_{true,GS}\|_2$ can be estimated as $p^N_{GS} \approx p^N_{LE}$. Notably, the global condition number is not involved in this case since the error source is the representation of the B.C. rather than the governing equation. Similar analysis can be done for $E^D_{\varphi,GS} = \|\boldsymbol{e}^D_{\varphi,GS}\|_2 / \|\boldsymbol{\varphi}_{true,GS}\|_2$. For calculations with Dirichlet B.C.s imposed directly on grid nodes, $\boldsymbol{b}^I_e = 0$ and $E^I_{\varphi,GS} = 0$.

The overall L_2 error for the velocity potential is

$$E_{\varphi,GS} = \frac{\|\boldsymbol{e}_{\varphi,GS}\|_2}{\|\boldsymbol{\varphi}_{true,GS}\|_2}, \tag{23}$$

with an upper bound $E_{\varphi,GS} \le E^I_{\varphi,GS} + E^D_{\varphi,GS} + E^N_{\varphi,GS}$. For a specific problem, its convergence order p_{GS} depends on the relative importance of $E^I_{\varphi,GS}$, $E^D_{\varphi,GS}$ and $E^N_{\varphi,GS}$, and generally p_{GS} lies somewhere between the minimum and maximum values of p^I_{GS}, p^D_{GS} and p^N_{GS}. According to the previous analysis in Section 2.2, the LE for the velocity potential inside the domain appears to be more accurate than that for the velocity if the same grid is used. Following from this, we can expect that $E^N_{\varphi,GS}$ and p^N_{GS} may dominate the global convergence rate unless we introduce some special treatment for Neumann boundaries to increase their accuracy. The next section will discuss different boundary treatments in detail. One should keep in mind that also other factors might be influential. Among these factors are e.g. the features of the true solution, conflicts between different B.C.s at intersection points of different boundaries, introduction of new nodes during grid refinement and accuracy in the geometrical description of boundaries. Moreover, the relative importance of different factors may change during grid refinement.

3.2 Assessment of global convergence rate through a test case

We use the shoebox problem in [1]. The computational domain is rectangular with $L = 40h$ (see Fig. 14). A global coordinate system *oxy* with origin located at the mid-point of the top surface is defined. The true solutions is taken as

$$\varphi_{true}(x,y) = \cosh\left[\bar{k}\left(y+h\right)\right]\sin\left(\bar{k}x\right). \tag{24}$$

This problem is equivalent to a boundary value problem for a linear free-surface wave in finite water depth *h*. The wave number $\bar{k}$ is chosen so that $\bar{k}h = 2\pi$. We define two B.C. cases: A case with pure Dirichlet B.C.s on all boundaries and a

mixed-B.C. case with Dirichlet B.C. at B.C.4 and Neumann B.C.s for the remaining boundaries B.C. 1-3. We use a squared-cell grid with $N_{HP} = N_{node} = 8$, and refine the grid homogenously to study the grid convergence of the global solution. The grid size here is measured by the ratio h/dx, where $dx = dy$ is the grid size. It should be noted that for the case with mixed B.C.s, conflicts between different B.C.s emerge at the domain corners. Such conflicts may represent a considerable error source. In the present calculations, this problem is avoided by using a higher-order cell with $N_{HP} = 9$ so that we can specify both conflicting B.C.s at corner nodes.

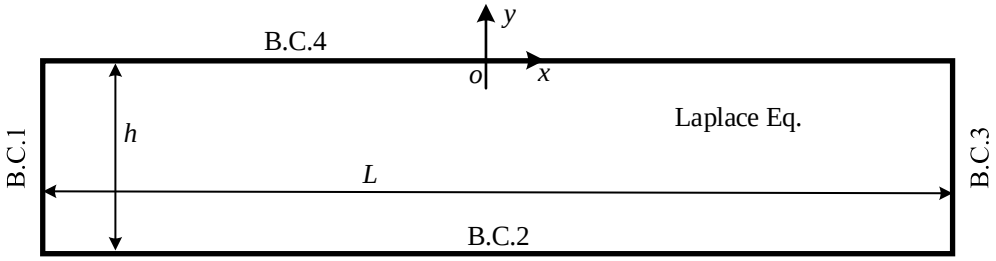
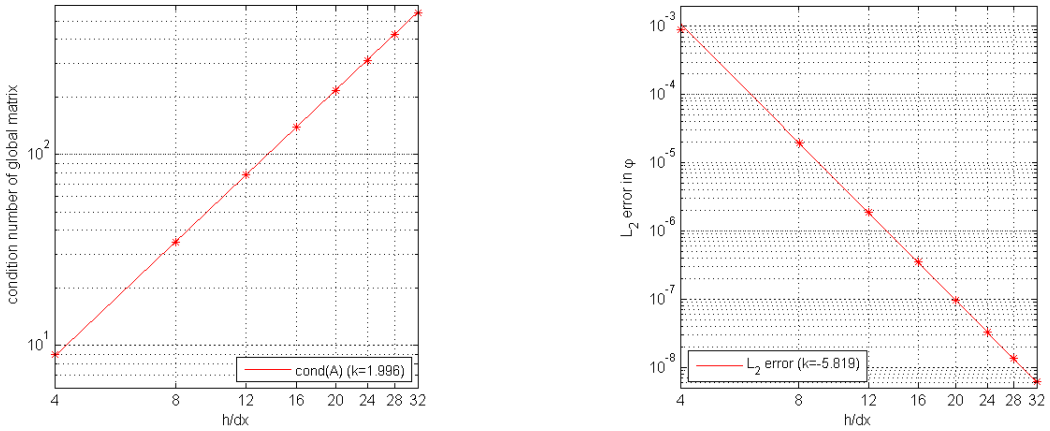


Fig. 14 Computational domain and boundary conditions for the shoebox problem

(1) Pure Dirichlet B.C.s case

In this case, the global error is solely due to discretization error inside the domain, i.e. $E_{\varphi,GS} = E^I_{\varphi,GS}$. According to Table 1, the squared-cell grid has 8th-order convergence for the velocity potential in the LE, i.e. $p^I_{LE} = 8$. Fig. 15a shows the condition numbers of the global coefficient matrix for different grid sizes, resulting in a convergence rate $p_{cond} \approx 2$. Thus, we expect the global convergence of the velocity potential to be $p_{GS} = p^I_{GS} \approx p^I_{LE} - p_{cond} \approx 6$. This is confirmed by Fig. 15b, where the L_2 error $E_{\varphi,GS}$ shows approximately 6th-order convergence.



(a) Condition number (b) Convergence of $E_{\varphi,GS}$

Fig. 15 Result for shoebox problem with pure Dirichlet B.C.s

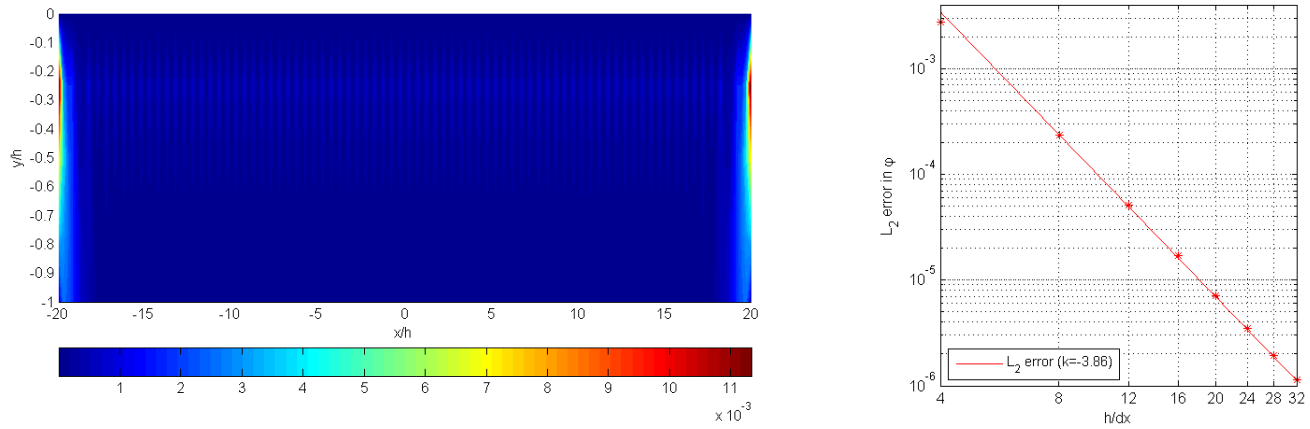
(2) Mixed Dirichlet-Neumann B.C.s case

This case is more complex due to contributions from the discretization errors not only inside the domain but also at the Neumann B.C.s. As an example, the relative error defined as

$$\boldsymbol{e}_{\varphi,GS,rel} = \frac{\left|\boldsymbol{e}_{\varphi,GS}\right|}{\max\left(\left|\boldsymbol{\varphi}_{true,GS}\right|\right)} \tag{25}$$

is contoured throughout the domain in Fig. 16a for a grid with $h/dx = 4$. Evidently, the errors are much larger near Neumann boundaries than elsewhere, which indicates that $\boldsymbol{b}^N_e$ in equation (20) is the main error source. Actually, almost 90% of the sum $\left(\left\|\boldsymbol{e}_{\varphi,GS}\right\|_2\right)^2$ is contributed from nodes within one wavelength $2\pi/k$ from B.C.1 and B.C.3. Thus, we expect the global

solution to converge nearly at the same rate as p_{LE}^{N}, which according to Table 1 is 4th-order for squared cells. This is also validated in Fig. 16b, where $E_{\varphi,GS}$ decreases at a rate close to 4th order during grid refinement.



(a) Contour of $e_{\varphi,GS,rel}$ when $h/dx = 4$ (b) Convergence of $E_{\varphi,GS}$

Fig. 16 Result for shoebox problem with mixed Dirichlet-Neumann B.C.s

3.3 Boundary Value Problem for the time derivative of the velocity potential

To estimate the hydrodynamic loads induced on bodies, within potential-flow assumption, the dynamic pressure

$$p = -\rho\frac{\partial\varphi}{\partial t} - \frac{1}{2}\rho|\nabla\varphi|^2 , \tag{26}$$

needs to be estimated along the body boundary [14]. In equation (26), the gradient of the velocity potential $\nabla\varphi$ comes directly from LE after the velocity potential has been determined. The $\partial\varphi/\partial t$-term cannot be determined directly from the velocity potential estimated at one time step and its wrong prediction can lead to force spurious oscillations in case of moving/deformable bodies crossing the grid cells. Hanssen et al. [15] used a higher-order backward FDM to determine $\partial\varphi/\partial t$ by taking the velocity potential at four subsequent time steps. It was found that this introduces small spurious oscillations in the pressure, also visible in the integrated pressure forces. In alternative, a separate boundary value problem (BVP) can be solved for $\partial\varphi/\partial t$ in order to prevent these oscillations. Several authors, e.g. [16-18], have successfully implemented this idea in BEMs, while this is the first time it is applied within a HPC formulation. For a general problem, the BVP for the acceleration potential $\partial\varphi/\partial t$ is:

$$\begin{cases} \nabla^2\left(\partial\varphi/\partial t\right) = 0 & in\ \Omega \\ \partial\varphi/\partial t = \partial\varphi_D/\partial t & on\ \partial\Omega_D \\ \dfrac{\partial}{\partial n}\left(\partial\varphi/\partial t\right) = \dfrac{\partial}{\partial t}\left(\dfrac{\partial\varphi_N}{\partial n}\right) - \nabla\varphi\cdot\dfrac{\partial \boldsymbol{n}}{\partial t} & on\ \partial\Omega_N \end{cases} , \tag{27}$$

where Ω is the computational domain, $\partial\Omega_D$ and $\partial\Omega_N$ are Dirichlet and Neumann boundaries, respectively. The BVP (27) for $\partial\varphi/\partial t$ can also be solved by the HPC method, and will have the same coefficient matrix as for the BVP for φ, i.e. matrix $\mathbf{A}$ in equation (16). We only have to replace the known vector $\boldsymbol{b}_{\varphi}$ for φ with the known vector $\boldsymbol{b}_{\partial\varphi/\partial t}$ for $\partial\varphi/\partial t$ in order to solve the new BVP. Thus, the generation and pre-conditioning of $\mathbf{A}$ need to be done only once for every time step, which means that the BVP for $\partial\varphi/\partial t$ leads to a limited increase of the computational effort, as confirmed by a test case in Section 4.

4. Modelling of generic geometries for rigid or deformable boundaries

Here three alternative local numerical treatments are proposed to improve the accuracy of the solution near the boundaries. They involve local refinement, use of higher-order cells, or combination of them. Then, three alternative grid strategies are

assessed for rigid or deformable boundaries with generic geometries. They correspond to a boundary-immersed grid, a boundary-fitted overlapping grid and a boundary-immersed overlapping grid.

4.1 Treatment for boundary condition

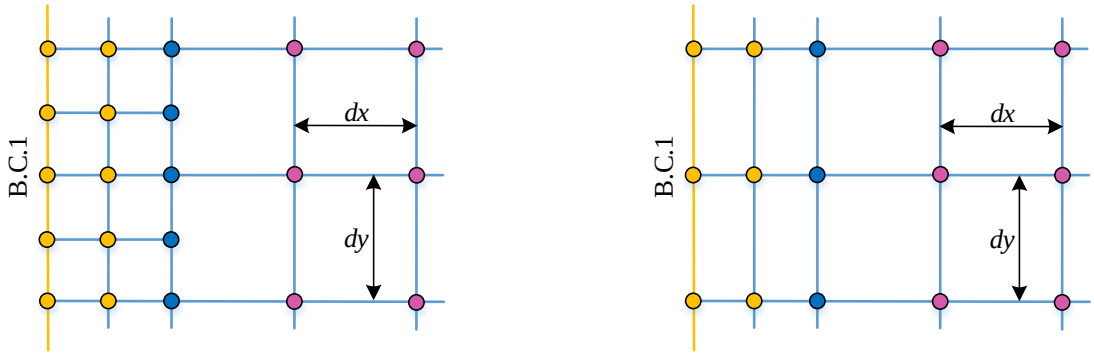
As illustrated in the last section, the errors due to boundary conditions can dominate the global accuracy, which can conceal the actual potentialities of the HPC method. Fortunately, we can mitigate this problem by introducing additional local treatments to reduce the errors from the B.C.s. Such treatments can be divided into three types: h-type, p-type and hp-type methods. The h-type methods reduce the grid size near boundaries, and can involve grid stretching or not, like the Quadtree or overlapping methods (see Section 2.2). The p-type methods use higher-order LEs for the boundary conditions, thus increasing the associated convergence rates. The hp-type methods combine the h-type and p-type treatments. In the following, we will discuss properties of these three methods using the shoebox problem in Fig. 14 as example. The cell with $N_{HP}=8$ and $N_{node}=8$ shown in Fig. 7a is still taken as basic order for the cells, with basic size $dx=dy$. According to the error distribution in Fig. 16a, it should be sufficient to apply the boundary treatments only for the Neumann boundaries at the left and right sides of the domain. To reduce complexity, no additional treatment is applied for the bottom Neumann boundary or the top Dirichlet boundary. The errors defined in equations (23) and (25) are used here and also in the rest parts of the paper.

(1) h-type method

For comparison purposes, two different kinds of grid refinement are carried out. In the first case, the grid is refined in both normal and tangential directions. As shown in Fig. 17a, the Quadtree grid method is applied. The "parent" nodes method introduced in Section 2.2.4 is adopted for the communication between grids. In the other case, the grid is only refined in normal direction to the boundary as shown in Fig. 17b. This is similar to what is commonly seen in CFD analyses. Fig. 18 shows the error $E_{\varphi,GS}$, the total number of nodes in the domain and the CPU time as a function of the grid density. In the figure, the "parent" grid size $dx=dy$ is used to calculate the grid density h/dx. "h-type-1" and "h-type-2" refer to using refinement in both directions simultaneously, within a distance dx and $2dx$ from the boundary, respectively. Thus, "h-type-1" is similar to Fig. 17a, while "h-type-2" has two more layers of "child" nodes. "h-type-1-x" refers to only using normal refinement within a distance dx from the boundary, as in Fig. 17b. Moreover, "original" refers to calculations without any boundary treatment, and "original-0.5dx" refers to the calculations without any boundary treatment but with half the grid size throughout the computational domain. In Fig. 18, dx is used to estimate the convergence rates; therefore, for a given h/dx in the figure, the value for "original-0.5dx" is equal to the value for "original" at $2h/dx$. Several findings emerge when comparing the different treatments:

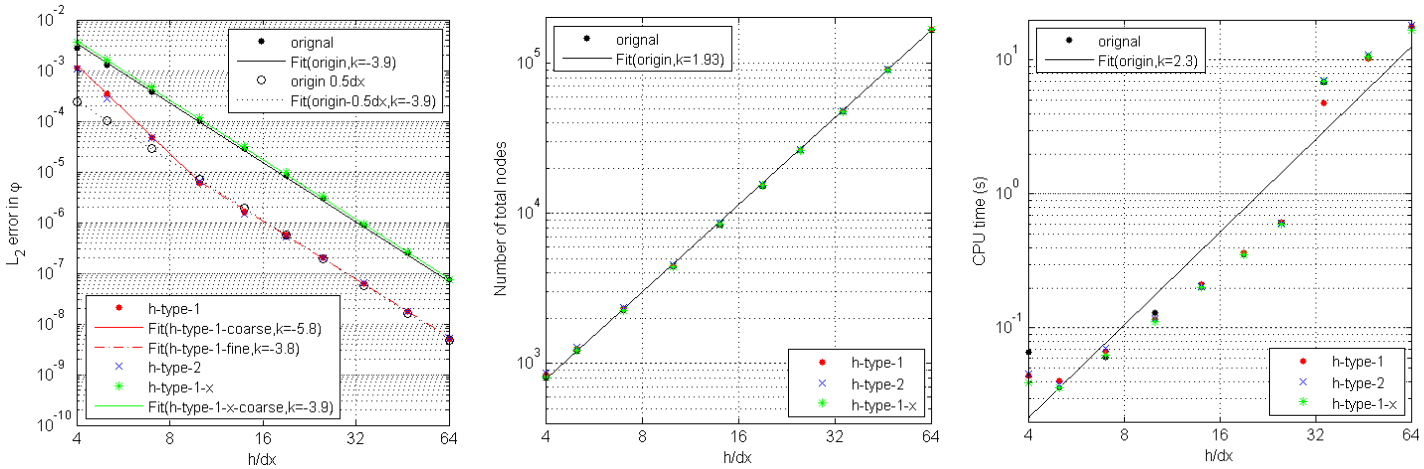
- Using refinement in both directions near Neumann boundaries is beneficial with respect to accuracy. By comparing the error $E_{\varphi,GS}$ for "h-type-1" and "original", the error is much smaller for the former. Moreover, when the grid is coarse, $E_{\varphi,GS}$ converges at approximately 6th order when using h-type treatment. This is because, after introducing the boundary treatment, the error from the LE inside the domain dominates the overall accuracy. This can be seen clearly from Fig. 19. For higher grid refinements, $E_{\varphi,GS}$ again converges at 4th order, and the errors are nearly the same as for the calculations using grid size $0.5dx$ throughout the domain. This is because the discretization error associated with Neumann boundaries dominates in this case.
- Using Quadtree grid for only one layer of the "parent" grid closest to the Neumann boundaries is sufficient. This is clear from the fact that the results for "h-type-2" are nearly the same as for "h-type-1".
- If the grid is refined, it should be refined in both normal and tangential direction. If refined only in normal direction as in "h-type-1-x", the errors are comparable or even slightly bigger than for "original". This observation agrees with the analysis in Section 2.2.2 regarding the error increase due to grid stretching.
- The h-type methods do not involve much extra computational effort, since the treatment is carried out locally. This is evident from Fig. 18b and c, in which the number of nodes and the CPU time for the different methods are nearly indistinguishable. It should be noted that the computations are carried out in MATLAB®, and a self-adaptive algorithm has been used to solve the global matrix equation (16). This explains why the CPU time of the finer grids has a different grid-size dependence with respect to the coarser grids.

It should be noted that the above findings are valid for h-type methods in general. Thus, similar observations can be expected if we substitute the Quadtree strategy with an overlapping-grid method. In fact, the Quadtree grid is a special kind of the overlapping-grid method where nodes in the two grids involved (i.e. parent and child) have common edges.



(a) Quadtree grid (b) Grid Refined only in normal direction

Fig. 17 Grids used in h-type methods (grids near B.C.1 in Fig. 14 for example)



(a) $E_{\varphi,GS}$ (b) Number of nodes (c) CPU time

Fig. 18 Grid convergence of error and computational efforts for h-type methods

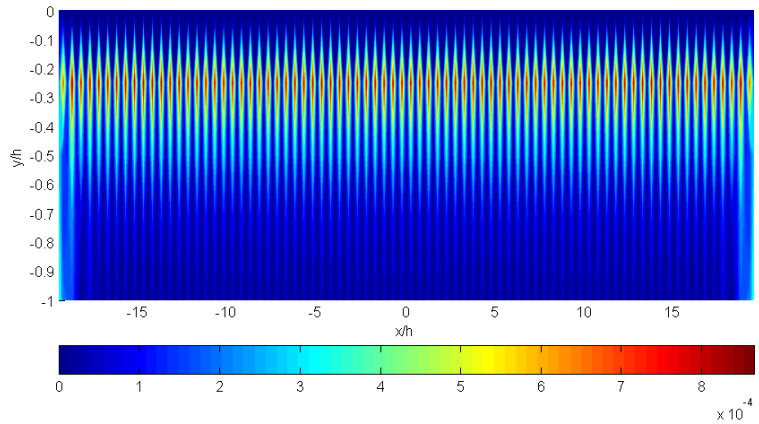


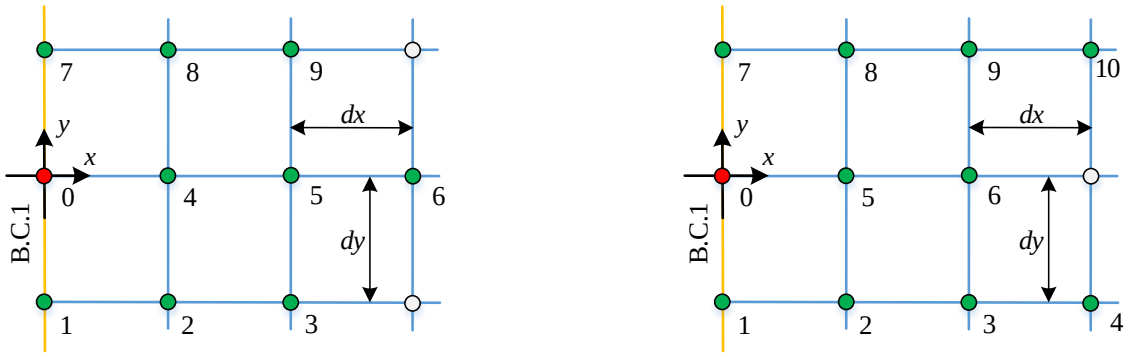
Fig. 19 Contour of $e_{\varphi,GS,rel}$ when $h/dx = 4$ using Quadtree grid

(2) p-type method

Fig. 20 shows two higher-order cells that can be applied for Neumann B.C.s to enhance the local accuracy with respect to the basic-order cell adopted inside the computational domain. The first one has $N_{HP} = N_{node} = 9$ with $\{h_j, j = 1 \sim 8,10\}$, while the second one has $N_{HP} = N_{node} = 10$ with $\{h_j, j = 1 \sim 10\}$. The choice of HPs follows from the discussion in Section 2.1.4.

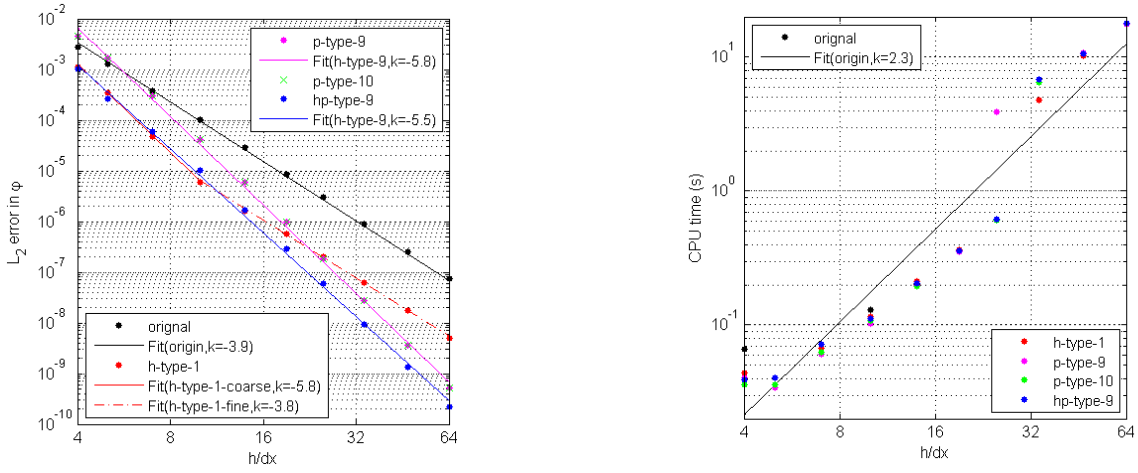
Due to the local symmetric node distribution, both associated LEs can achieve 5th-order accuracy for the normal velocity. All the simulations with p-type treatment use uniform grids, i.e. without local refinement. Fig. 21 shows the error $E_{\varphi,GS}$ and the CPU time as a function of the grid density. "p-type-9" and "p-type-10" refer to the two different higher-order cells in Fig. 20. The findings from the analysis are as follows:

- The high-order LEs for Neumann B.C.s significantly improve the convergence of the error. The "p-type-9" has a convergence rate between 5th and 6th order, since discretization errors due to both Neumann B.C.s and nodes inside the domain are important.
- The 9-node cell can achieve similar error values as the 10-node cell. Although the latter can have better accuracy for the velocity potential, their accuracies for normal velocity are nearly the same. Thus, it is generally sufficient to use the 9-node cell for Neumann B.C. treatment.
- When the grid is coarse, the h-type treatment can give smaller error than the p-type methods. However, when the grid is fine, the p-type methods will be more accurate due to their higher-order convergence properties.
- As shown in Fig. 21.b, the p-type method will not significantly increase the computational effort. This is because the treatment is applied locally near Neumann boundaries, and the bandwidth of the global matrix does not increase notably.



(a) 9-node cell (b) 10-node cell

Fig. 20 Cells used in p-type methods (cells near B.C.1 in Fig. 14 for example)



(a) $E_{\varphi,GS}$ (b) CPU time

Fig. 21 Grid convergence of error and computational efforts for p-type and hp-type methods

(3) hp-type method

From the above analysis, the h-type and p-type treatments are almost complementary. Therefore, it is worth trying to combine their features for an overall optimized accuracy. The hp-type treatment used to achieve this applies the Quadtree grid as shown in Fig. 17a combined with a 9-node cell for Neumann B.C.s. The associated error $E_{\varphi,GS}$ and CPU time for this case, denoted as "hp-type-9", are plotted as a function of grid density in Fig. 21. The following observations are made:

- The hp-type treatment can achieve small errors similar to the h-type treatment when the grid is coarse, while maintaining a high converge rate between 5^{th} and 6^{th} order similar to the p-type treatment.
- The applied hp-treatment does not significantly increase the computational effort.

Thus, the hp-type treatment is recommended to reduce numerical errors associated with Neumann B.C.s in general. It should be noted that the techniques discussed above are not restricted to boundary treatment. In fact, they can also be used to reduce general error sources, e.g. the interpolations between overlapped grids.

4.2 Treatment for generic boundary geometry and dynamic boundaries

For the shoebox problem described above, the computational domain is rectangular and easily discretized with squared cells. However, most practical applications involve irregular boundary geometries. The situation can be even more complicated when potential-flow time-domain problems are studied and dynamic boundaries are involved, implying that the domain boundaries change position or geometry in time. In such cases, surface tracking or capturing methods are necessary for these boundaries. Shao and Faltinsen [1-3] used boundary-fitted structured grids for both the free-surface and the body-surface. In such analyses, either the entire or parts of the grid have to be updated at every time step. This requires recalculations associated with the LEs. The results shown in their papers generally have good agreement when compared to experiments and other numerical solutions. However, a general limitation for this type of boundary-tracking method is that it is hard to conserve the quality of structured grids for complicated boundary geometries and/or large boundary deformations. In some cases, it will be impossible to avoid severe grid stretching or distortion. According to the analysis in Section 2, this may significantly degenerate the accuracy of the HPC method. In order to avoid this limitation, Hanssen et al [15] proposed an immersed boundary method for moving bodies. In their work, a fixed squared grid is applied and the body-surface geometry is tracked by moving markers. Three layers of ghost nodes were used inside the body to improve the estimates of the $-\rho\partial\varphi/\partial t$ -pressure term in the Bernoulli equation by a Finite Difference Method (FDM). For every ghost node, three mirror points were defined in the fluid and used to establish an interpolation scheme to satisfy the zero-penetration boundary condition on the body-surface. A disadvantage of this method is that the treatment of Neumann boundary conditions become complicated and may introduce spurious pressure oscillations due to the FDM. Furthermore, it assumes implicitly that analytical continuation of the solution to inside the body is possible. Other boundary capturing or tracking methods such as level-set and volume-of-fluid may also be applicable, see e.g. [19, 20]. When choosing a boundary treatment, one should take into account the properties of the HPC as discussed in Section 2 and 3. The following suggestions can be helpful in doing this selection:

- The method should avoid modification of too many cells during time marching. As shown in Section 2.1 and 2.2.2, the LEs based on HPs are not conserved for all affine transformations. This means that the local matrix inversion may have to be re-performed after modification of a cell. Although the size of the local matrix to be inverted is usually small, it may still accumulate significant CPU time if the inversion has to be performed for many cells. In order to avoid such additional contributions to the CPU-time cost, the relative position of most of the grid nodes in the domain should be preserved between computational steps.
- The method should seek to avoid additional large error sources. This is especially important since the HPC method is a method with high-order accuracy, and the error of the global solution can be very sensitive to local treatments.
- The method should use squared/symmetric cells as far as possible. As already discussed in Section 2.2.1, these cells generally give superior accuracy.
- The method should consider the local distribution of errors within cells. This is important for methods involving immersed boundaries or overlapping grids, since one has to choose a proper HPC cell for interpolation. The distribution of errors within cells has been discussed in Section 2.2.3. As a general advice, if the cell in Fig. 7a ($N_{HP}=N_{node}=8$) is chosen, one should make sure that $|x_{0c}|\le 0.5dx$ and $|y_{0c}|\le 0.5dy$. This means that the interpolation point should be within half the grid size from the cell center. The cells in Fig. 7b and Fig. 7c are accurate over a larger area within the cells, but are also more complicated to implement. Moreover, one should be cautious when estimating velocities, since the error distribution for velocities is different from that of the potential.

With background in the above recommendations, we propose three different boundary treatments in the present paper:

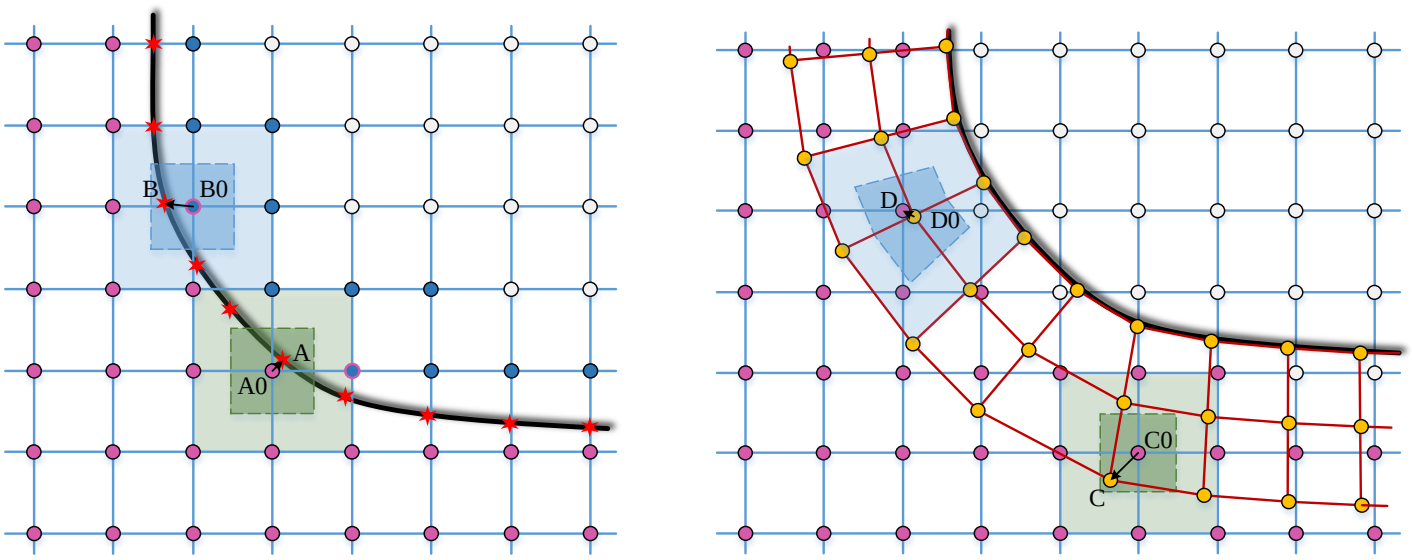
(1) Immersed Boundary Grid (IBG)

This method is modified version of the one proposed by Hanssen et al [15]. The main difference is in the identification of ghost nodes and treatment of boundary conditions. Fig. 22a shows a principal outline of the IBG, where the black curve is

the boundary with the physical computational domain on the left-bottom side. The purple circles are nodes in the physical domain, the blue circles are ghost nodes outside the physical domain, the white circles are inactive nodes outside the physical domain that are not involved in the computation and the red stars are markers on the boundary. If the boundary location or geometry is updated, the node types should be identified again and the markers should be redefined. In order to save CPU time, we only need to change the node type for nodes close to the boundary at the previous time step, since the CFL number limits the change per time step in practice. The boundary condition is satisfied at the marker locations. For every marker, a HPC cell should be chosen and its LE gives the velocity potential or velocity at the marker position. Selection of HPC cells is done in accordance with previous recommendations. As shown in Fig. 22a, taking a $N_{HP} = N_{node} = 8$ cell as example, the cell (green shaded) centered at node A0 should be chosen for marker A, since marker A is within its 'safe zone' (darker green shaded, satisfies $|x_{0c}| \le 0.5dx$ and $|y_{0c}| \le 0.5dy$) . All nodes involved in the chosen cell should be either ghost nodes or nodes in the physical domain. Similar treatment applies for marker B, where the cell (blue shaded) centered at B0 also involves ghost nodes not directly adjacent to the boundary. Moreover, since the ghost node B0 is surrounded by either ghost nodes or nodes inside the domain, by continuation of the solution, its corresponding equation is (7) as for other nodes inside the domain. In order to achieve a solvable global problem, the number of unknowns should be equal to the number of equations. Thus, the number of markers should be the same as the number of ghost nodes surrounded by at least one inactive node, and every ghost node should be involved in at least one of the LEs in use. The identification of ghost nodes and distribution of applied markers influence each other. Typically, the accuracy of the final solution will depend not only on the distance between markers and cell centers in the selected cells for the LEs, but also on how evenly spaced are the markers selected for boundary conditions. These two factors may conflict, and an optimization of the method, invoked to compromise between them, can improve the accuracy.

(2) Boundary-fitted Overlapping Grid (BFOG)

This method uses a squared background grid for most of the domain, with a boundary-fitted structured grid for the area in vicinity of the boundary. The different grids should have two-way communication based on HPC interpolation. If the boundary location or geometry is changed, the boundary-fitted grid is updated accordingly while the background grid remains unchanged. Thus, the communication between the grids must also be re-established. Fig. 22b shows a principal outline of the BFOG scheme, with similar definitions of the physical domain and boundary as in Fig. 22a. Blue lines belong to the background grid, with purple and white circles for active and inactive nodes, respectively. The red lines belong to the boundary-fitted grid, with yellow circles indicate the corresponding grid nodes. Boundary conditions are applied on the first layer of boundary-fitted nodes belonging to the boundary. The velocity potential at the innermost (i.e. farthest from the boundary) node layer in the boundary-fitted grid is interpolated from the LEs of suitable cells in the background grid. Similar considerations, as those made for the IBG method, should be taken when choosing the cell for each node. Using node C in the boundary-fitted grid as an example, we choose the background-grid cell (green shaded) centered at node C0 for interpolation. All the background-grid nodes included in this cell must be active. If an active node in the background grid is surrounded by at least one inactive node, its solution should be interpolated from the LE of a suitable cell in the boundary-fitted grid. As an example, the boundary-fitted grid cell (blue shaded) centered at D0 is used to interpolate the solution for the background-grid node D. In order to ensure this two-way communication, the boundary-fitted grid should have at least three radial layer of nodes. Increasing the number of node layers further gives increased flexibility but also increases the computational effort.



(a) Immersed boundary grid (b) Boundary-fitted overlapping grid

Fig. 22 Grid concepts implemented for boundaries

(3) Immersed Boundary Overlapping Grid (IBOG)

This method is a combination of the IBG and BFOG methods. There are two set of grids, one is an Earth-fixed squared background grid, and the other is a boundary-fixed immersed-boundary grid moving together with the boundary. Two-way communication is established between the two grids in a way analogous to the BFOG method. The difference between the IBOG and BFOG methods is that the boundary-fixed grid in the IBOG method is also squared, and uses markers and ghost nodes in a similar way as the IBG method to capture the boundary.

It should be noted that it is common to precondition the matrix before solving the problem. In the original HPC method presented by Shao and Faltinsen [1-3], the structure of the matrix remains the same. Thus in most of cases, the matrix was preconditioned only once and then used in the subsequent time-domain analysis. This property can significantly save CPU time. The three grid methods introduced above usually involve changes of matrix structure during the calculations. This is mainly because some of the background nodes will turn from inactive to active or vice versa. One possible solution for this problem may be including more background nodes (regardless that some of them are actually inactive for a certain instant), so that the number of unknowns will not be changed due to boundary motions. The inactive nodes can be set with Dirichlet condition and equal to an arbitrary finite value. Another solution may be using the multigrid approach for preconditioning, as done by Engsig-Karup et al [21]. This preconditioning issue will be studied as a future work, and for the following calculations, preconditioning is carried out before every time step.

In addition to the discussions in Section 3, special attention is required for the error sources involved when using a certain grid treatment. For the IBG and IBOG methods, involving Dirichlet B.C.s applied on markers rather than grid nodes, $\boldsymbol{e}^{D}_{\varphi,GS}$ is usually non-zero. For the BFOG and IBOG methods, the grid-grid communication can introduce additional error sources, which can also be interpreted as a kind of $\boldsymbol{e}^{D}_{\varphi,GS}$ for each grid separately.

In the following, the global properties and the accuracy of the HPC method, with the different grid strategies described above, are examined for three example problems. Firstly, all three method above have been implemented for the potential flow around a submerged circular cylinder. This problem has an analytical solution and has also been considered by Hanssen et al. [15]. Secondly, the IBG and BFOG methods are implemented to simulate the potential flow in a nonlinear numerical wave tank with periodic boundary conditions. As for true solution, this problem has a highly-accurate approximate solution represented by a series of Fourier components given by Rienecker and Fenton [22]. At last, the IBOG method is implemented to a heaving circular cylinder semi-submerged in still water, and the results are compared with those by Sun [23] using fully nonlinear BEM. Together these three examples represent BVPs with both rigid (body) and deformable (free surface) boundaries, which are typical scenarios in marine engineering problems.

4.2.1 Example 1: potential flow around a rigid body in infinite fluid

Fig. 23 illustrates the BVP for a fully submerged 2D circular cylinder in infinite fluid. We assume potential flow, thereby neglecting viscous effects. A squared computational domain with $l_x = l_y = 4D = 40m$ is chosen, with the cylinder initially located in the domain center. The analytical solution is specified as Dirichlet condition ($\varphi = \varphi_{true}$) on the four outer boundaries. A zero-penetration condition must be satisfied on the body-surface, i.e. the normal fluid velocity at the boundary is set equal to the normal velocity of the boundary itself ($\varphi_n = \boldsymbol{U}_{cyl.} \cdot \boldsymbol{n}$). This scenario is equivalent to that studied in [15]. An Earth-fixed coordinate system *oxy* is defined in Fig. 23, with origin o coinciding with the initial position of the cylinder center.

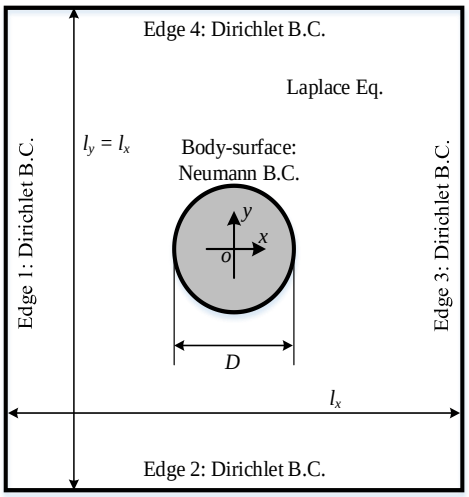


Fig. 23 Computational domain and boundary conditions for potential flow around a submerged cylinder

Initially, we perform calculations for a steady case with the cylinder fixed. A uniform inflow in *x* direction with velocity $U_0 = 1m/s$ is specified, with the corresponding analytical solution given as:

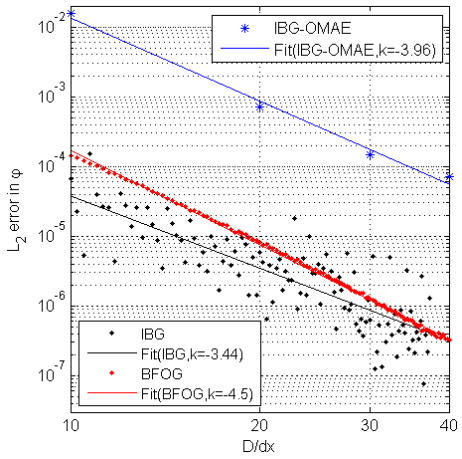
$$\varphi_{true}(x,y) = U_0\left(x + \frac{x}{x^2+y^2}\frac{D^2}{4}\right). \tag{28}$$

We solve the BVP numerically with the IBG and BFOG methods, varying the grid size D/dx from approximately 10 to 40. For the IBG method, we apply an uniform, structured grid. The markers applied in the body-boundary condition are distributed so that their distances to the center of the cells selected for the LEs are small. For the BFOG method, the uniform, structured background grid has grid size dx. The body-fitted grid has five radial layers of nodes separated by dx, which is equal to the tangential node separation in the innermost (i.e. farthest from the boundary) layer. Due to the curvature of the cylinder surface, the tangential node separation on the cylinder boundary will be less than dx, which implies a modest grid stretching. We use the $N_{HP} = N_{node} = 8$ cell in Fig. 7a to form the LE for body markers in the IBG method and for the grid-grid communications in the BFOG method. Fig. 24a shows the L_2 norm of the error in velocity potential calculated by both methods. The plot also includes the errors from [15], denoted as 'IBG-OMAE', for comparison. The IBG method is implemented in a Python code, while the BFOG method is programmed in FORTRAN. This means that it would not be strictly correct to compare the CPU time of the two methods directly. A more sound way to indicate their respective computation efforts in this case is to consider the total number of active nodes used in both methods. Fig. 24b shows this, where the number of active nodes dictates the dimensions of the coefficient matrix $\mathbf{A}$ in equation (16). This is the governing factor influencing the computation effort involved in solving the linear system of equations resulting from the BVP. The figures show that:

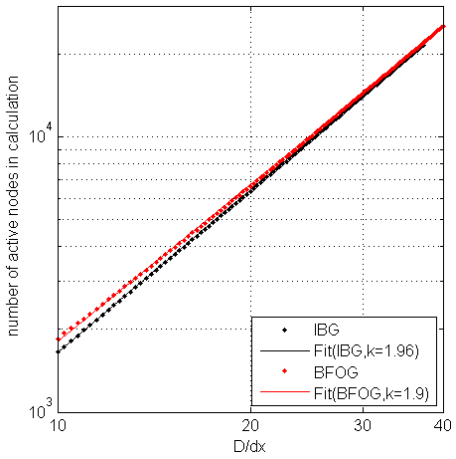
- The IBG method adopted in this paper has a significantly lower error than the one in [15]. This is mainly because the Neumann B.C. on the body-surface is expressed directly by LEs based on HPs in this paper, which is expected to be a more accurate scheme than the interpolation scheme used in [15].

- The error in the IBG method shows significant oscillations during grid refinement. This is mainly because the distribution of markers used in the body-boundary conditions changes when the grid is changed. For two different grid densities, the distance between markers and corresponding cell centers changes, as does the relative distance between adjacent markers. This can be explained by the fact that it is difficult to optimize both these parameters at the same time, so that the situation for some grids becomes very favorable while it is less favorable for others. Another factor that can give some oscillations is that only nodes outside the cylinder are considered when computing the error by (23). We expect the oscillations due to this to be smaller than those caused by the distribution of markers. One should anyway be aware of this oscillation source when using IBG-based methods. In particular, a large number of data points are required in grid convergence studies in order to reveal the oscillatory-convergence behavior. Too few points may give false conclusions regarding the convergence rate.
  The errors for the IBG and BFOG methods are of similar magnitude when the grids are refined, while the former can have smaller errors for coarse grids. Fig. 25 shows the spatial relative errors $e_{\varphi,GS,rel}$ defined by (25) in vicinity of the body-surface for the case $D/dx = 11.5$. As shown in Fig. 25a, the largest $e_{\varphi,GS,rel}$ in the IBG method concentrates locally in some regions near the body-surface, which relates to the previous discussion regarding the influence of marker distribution. Fig. 25b shows that $e_{\varphi,GS,rel}$ in the BFOG method has large values within a relatively large region. This is due to stretching of the boundary-fitted grid, which may reduce the accuracy of the LE. This stretching becomes more profound near the cylinder surface. Moreover, the maximum magnitude of $e_{\varphi,GS,rel}$ in the BFOG method is larger than that in the IBG method.
- The error in the BFOG method converges without visible oscillations at a rate of approximately 4.5. This is in accordance with the discussion in Section 2.2 and 3. In the boundary-fitted grid with stretched but symmetric cells, the Neumann B.C. at the body-surface gives 4th-order convergence for both the LE and $E^N_{\varphi,GS}$. The potential inside the domain has 5th-order convergence for the LE, which corresponds to a 3rd-order convergence for $E^I_{\varphi,GS}$. In the background grid with squared cells, 8th-order convergence can be achieved for the LE, which corresponds to 6th-order convergence for $E^I_{\varphi,GS}$. The communication between the grids involves 5th-order or better convergence. Thus, the global error $E_{\varphi,GS}$ in the BFOG method will have between 3rd- and 6th-order convergence. A similar analysis for the IBG method tells us that it has a 3rd-order convergence for $E^N_{\varphi,GS}$ and a 6th-order convergence for $E^I_{\varphi,GS}$. This indicates that the global error $E_{\varphi,GS}$ has between 3rd- and 6th-order convergence also for this case. Due to the oscillatory convergence of the error in the IBG method seen in Fig. 24a, it is difficult to compare its convergence rate directly with the BFOG method.
- The number of total active nodes involved in the computation is rather similar for the IBG and BFOG methods. The latter will generally require some more nodes associated with the boundary-fitted grid, but the difference diminishes as the grid density is increased.

It is possible to improve the accuracy of both the IBG and the BFOG methods. For example, the treatments introduced in Section 4.1 can be used to reduce the error at the Neumann boundaries. Moreover, it may also be helpful to use the higher order cells in Section 2.2.3 for the LEs used for the markers in the IBG method and for the grid-grid communication in the BFOG method. It is also likely that such treatments will reduce the oscillatory-convergence behavior for the IBG method. At last, it should be noted that the ghost nodes in immersed boundary methods require analytical continuation of the solution outside the fluid domain, where a singularity may happen near a boundary with high local curvature. One solution for this problem is local refinement of the grid, e.g. the Quadtree grid introduced in Section 2.2.4 and 4.1..

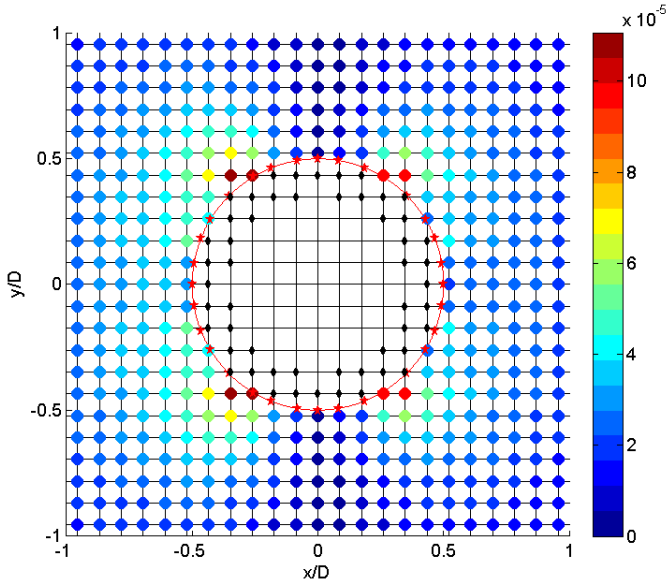


(a) $E_{\varphi,GS}$

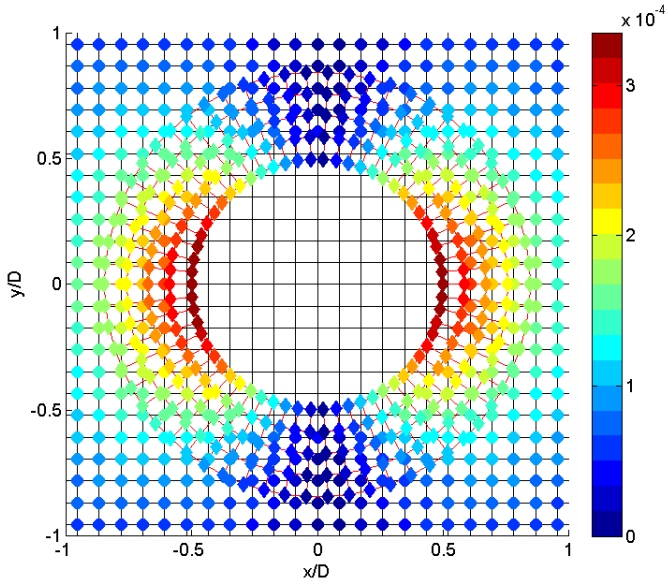


(b) Number of total active nodes

Fig. 24 Grid convergence of $E_{\varphi,GS}$ and number of total active nodes for submerged cylinder example



(a) Immersed boundary grid
(color filled circles for nodes in domain, small black filled diamonds for ghost nodes in use, and red stars for markers)



(b) Boundary-fitted overlapping grid
(color filled circles for background nodes, color filled diamonds for boundary-fitted nodes)

Fig. 25 Distribution of $e_{\varphi,GS,rel}$ in vicinity of cylinder ($D/dx=11.5$)

In the following, the circular cylinder is studied as an oscillating body in otherwise still fluid. The cylinder is forced to oscillate in x direction with a velocity amplitude $U_a=1m/s$ and oscillation frequency $\omega=0.5rad/s$. The forced motion of the cylinder center is

$$\left(x_{cyl},y_{cyl}\right)=\frac{U_a}{\omega}\left(\sin\omega t,0\right). \tag{29}$$

The corresponding analytical solution of the velocity potential for this case is

$$\varphi_{true}\left(x,y\right)=-U_a\cos\omega t\left(\frac{\left(x-x_{cyl}\right)}{\left(x-x_{cyl}\right)^2+\left(y-y_{cyl}\right)^2}\frac{D^2}{4}\right). \tag{30}$$

The IBG, IBOG and BFOG methods are used to simulate the flow around the oscillating cylinder for squared grids with grid sizes $D/dx=10,13,16,\cdots 40$. A similar implementation for the IBG and BFOG methods as for the steady case is used. For the IBOG method, the boundary-fixed grid covers a squared region defined by $\left|x-x_{cyl}\right|\le D$ and $\left|y-y_{cyl}\right|\le D$, and has

the same grid size as the background grid. The communication between the two grids uses the $N_{HP} = N_{node} = 8$ cell in Fig. 7a. In the following, the discussion focuses on the force acting on the moving cylinder, as this is often a quantity of practical interest. The force is determined by directly integrating the pressure around the cylinder surface. Omitting the hydrostatic pressure, the dynamic pressure follows from Bernoulli's equation (26) with $\rho = 1000 kg/m^3$ as fluid density once both the velocity potential and its time derivative have been found. For $\partial\varphi/\partial t$, the BVP (27) can be solved as discussed in Section 3.3. For the rigid cylinder moving according to law (29), $\partial(\partial\varphi/\partial t)/\partial n = -\omega U_a \sin\omega t \cdot \cos\theta$ on the body boundary, with $\cos\theta = (x - x_{cyl}) \Big/ \sqrt{(x - x_{cyl})^2 + (y - y_{cyl})^2}$. Fig. 26 provides the CPU-time per time step for the BFOG method as a function of grid density with $\partial\varphi/\partial t$ determined by either a FDM or directly from the BVP (27). The results confirm a limited increase of the computational effort connected with the solution of the BVP for $\partial\varphi/\partial t$. Moreover, the FDM generally requires a smaller time step to ensure accurate determination of $\partial\varphi/\partial t$, which means that the BVP for $\partial\varphi/\partial t$ in practice can represent the most efficient option. In the present study, the time step is fixed so that $dt/T = 40$ for the computations and $\partial\varphi/\partial t$ is found from a BVP. In this case, the time step value does not have a direct influence on the accuracy of $\partial\varphi/\partial t$, and it is chosen to resolve time series with enough detail.

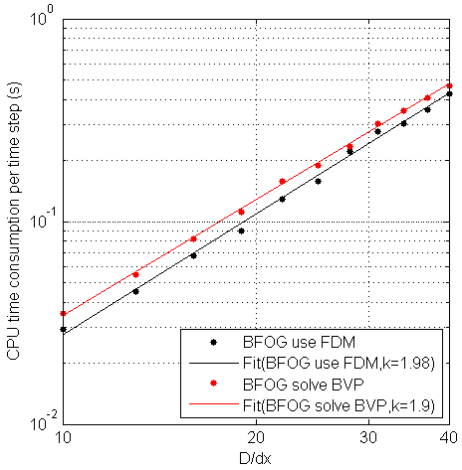


Fig. 26 CPU time consumption per time step by different methods to achieve $\partial\varphi/\partial t$

One of the advantages of an overlapping grid is that, for dynamic boundaries that do not deform, a boundary-fixed local reference frame can be used for the grid fixed to a dynamic boundary. This enables us to describe the flow around a body more conveniently, and solutions are even possible for BVPs with singularities due to sharp corners [24]. Moreover, in this local boundary-fixed reference frame, $\partial \boldsymbol{n}/\partial t = (0,0)$ and will vanish for the Neumann B.C. in BVP (27). Coordinate transformations are necessary for the two-way communications between grids described by different coordinate systems. Let us consider a boundary-fixed coordinate system translated by (x_b, y_b) and rotated by β_b with respect to the Earth-fixed coordinate system. We define $\tilde{\varphi}(\tilde{x}, \tilde{y})$ as the potential in a point in the body-fixed reference frame and it coincides with $\varphi(x, y)$ in the Earth-fixed reference frame. The coordinates in the two reference frames are related through

$$\begin{cases} x = \tilde{x}\cos\beta_b - \tilde{y}\sin\beta_b + x_b \\ y = \tilde{x}\sin\beta_b + \tilde{y}\cos\beta_b + y_b \end{cases}. \tag{31}$$

The formulations for $\varphi(x, y)$ and $\partial\varphi(x, y)/\partial t$ in Earth-fixed system can be transformed to boundary-fixed system as follows

$$\varphi(x, y) = \varphi\big(x(\tilde{x}, \tilde{y}), y(\tilde{x}, \tilde{y})\big) = \tilde{\varphi}(\tilde{x}, \tilde{y}), \tag{32}$$

$$\frac{\partial}{\partial t}\varphi(x, y) = \frac{\partial}{\partial t}\varphi\big(x(\tilde{x}, \tilde{y}), y(\tilde{x}, \tilde{y})\big) = \frac{\partial}{\partial t}\tilde{\varphi}(\tilde{x}, \tilde{y}) - \boldsymbol{v}_b \cdot \nabla\varphi(x, y), \tag{33}$$

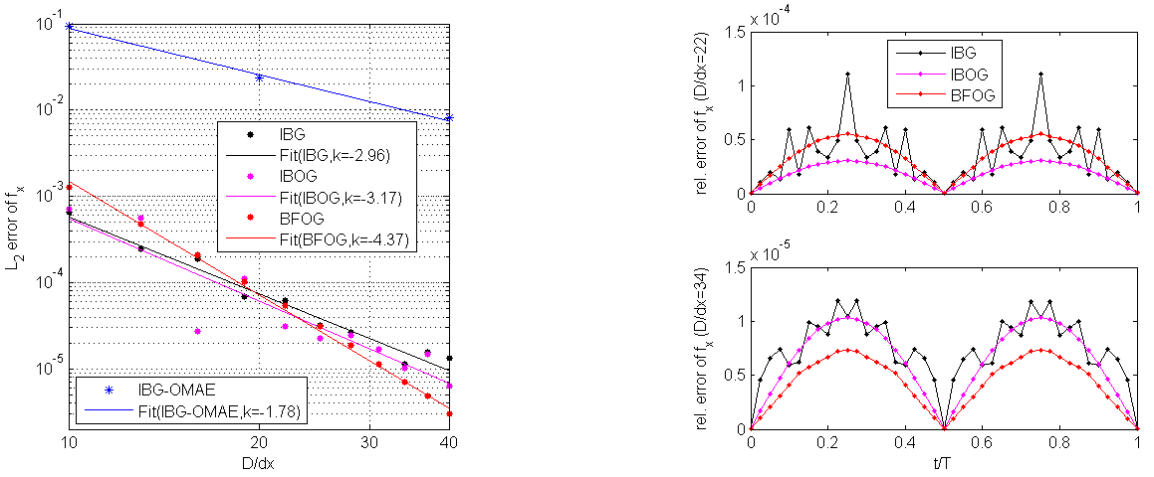
with

$$\boldsymbol{v}_b = \left(\frac{\partial x_b}{\partial t} - \left(\tilde{x} \sin\beta_b + \tilde{y} \cos\beta_b \right) \frac{\partial \beta_b}{\partial t},\ \frac{\partial y_b}{\partial t} + \left(\tilde{x} \cos\beta_b - \tilde{y} \sin\beta_b \right) \frac{\partial \beta_b}{\partial t} \right).$$

The $\partial\tilde{\varphi}(\tilde{x},\tilde{y})/\partial t$ in equation (33) is defined in the boundary-fixed reference frame. Similar formulations apply for the inverse transformation. It should be noted that the convective term in equation (33) involves the velocity vector $\nabla\varphi(x,y)$, which, according to Section 2.2, may reduce the order of accuracy. If needed, this negative effect can be neutralized by using local treatments introduced in Section 4.1.

Due to the symmetric feature of the flow for the chosen case, the force component in *y* direction f_y is negligible for all the simulated cases, which agrees well with the analytical value of zero. The analysis will thus focus on the in-line force f_x in *x* direction. The L_2 error $E_{f_x,GS}$ of f_x is defined similarly to the error (23). Here the summation in $\|\ \|_2$ is carried out as a time integration, which means summation of values at each time step over one oscillation period $T = 2\pi/\omega$. Fig. 27a shows the obtained $E_{f_x,GS}$ as a function of grid density, and also includes the results from [15] for comparison. Fig. 27b shows the variation of the error in f_x over one oscillation period for grid size $D/dx = 22.34$. It is clear that:

- All three methods used in the present paper give smaller errors and faster grid convergence than that in [15]. The reason for this is three fold: As shown in Fig. 24a, the accuracy of the velocity potential is improved in the present paper. Secondly, the BVP for $\partial\varphi/\partial t$ is more accurate than the FDM used in [15]. Thirdly, the pressure integration scheme was less accurate in [15].
- The errors for the IBG and IBOG methods still show some oscillatory convergence during grid refinement. For the IBOG method, the behavior is consistent with the oscillations in $E_{\varphi,GS}$ seen in Fig. 24a. For the IBG method, the trend is more complicated since the ghost nodes and markers used in the body-boundary condition will change as the cylinder moves.
- The error for the BFOG method seems to convergence faster than the two methods with immersed boundaries. However, these methods can give smaller error magnitudes for coarse grids. We can recognize this from the behavior of the error in velocity potential in Fig. 24a.
- As shown in Fig. 27b, the IBG method shows spurious oscillation in the error time series of f_x. As touched upon above, this is mainly because the ghost nodes and applied body markers change during the simulation. The figure also shows that both overlapping-grid methods effectively suppress these spurious oscillations.



(a) L_2 error of f_x (b) history of $e_{f_x,rel}$ ($D/dx = 22,\ 34$)

Fig. 27 Error in f_x for different grid methods

4.2.2 Example 2: potential flow in a nonlinear wave tank with periodic boundary conditions

Fig. 28 shows a sketch of the Earth-fixed wave tank with length l_x equal to one wavelength $\lambda = 10m$ and a water depth $h = \lambda/2$. A Cartesian coordinate system oxy is used with the origin o in the middle of the tank at the mean water level. The examined problem assumes waves propagating in the tank under steady-state conditions. According to [14], there will be a mass transport caused by the Stokes-drift velocity. In our simulation, this effect is considered by assuming that the velocity potential throughout the tank will have a linear decrease with time. Under these assumptions, spatial periodic B.C. $\varphi_{edge1} = \varphi_{edge3}$ applies for the velocity potential at the two vertical domain boundaries. A zero-penetration Neumann B.C. $\partial\varphi/\partial n = 0$ is imposed on the seabed. Because the wave tank is fully-nonlinear, the free-surface deforms during the simulation. At a certain time step, the BVP is established by imposing the velocity potential as a Dirichlet B.C. $\varphi = \varphi_{fs}$ on the exact free-surface elevation η. The kinematic and dynamic free-surface boundary conditions are used to march η and φ_{fs} in time. As reference solution, we use the method derived by Rienecker and Fenton [22] through expanding the stream function in a Fourier series. The wave steepness is set to be $ka = kH/2 = 0.1$. The first 20 Fourier components are used to approximate the analytical solution; this has been confirmed to be sufficient.

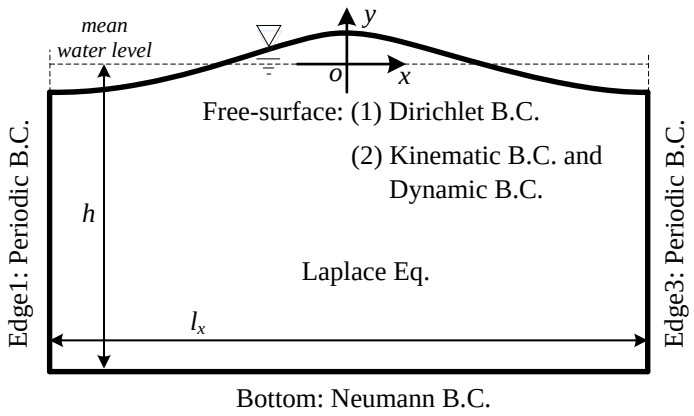


Fig. 28 Computational domain and boundary conditions for a nonlinear wave tank with periodic edges

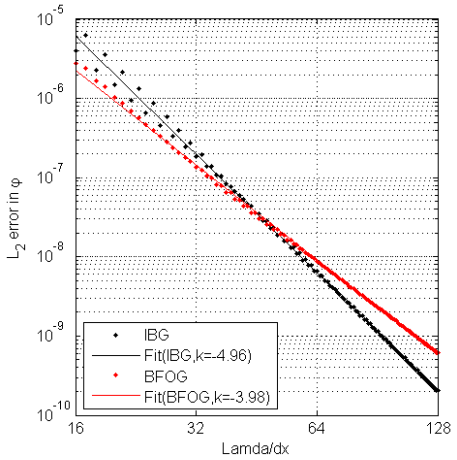
Firstly, we consider a BVP at the initial time $t = 0$ with the wave crest in the middle of the tank. The free-surface elevation and potential on the free-surface are taken from the analytical solution, and the BVP is solved with the IBG and BFOG methods for grid densities λ/dx in the range 16-128. For the IBG method, the grid is structured and uniform throughout with one marker located on every vertical grid line. The markers are restricted to move only in vertical direction, i.e. the free surface can only be a single-valued function of x. For the BFOG method, the background grid is structured and uniform with grid size dx. The free-surface-fitted grid has five radial node layers, with nodes separated by a distance dx in both directions. The $N_{HP} = N_{node} = 8$ cell in Fig. 7a is used in the LEs for the free-surface markers in the IBG method and in the grid-grid communication in the BFOG method. Fig. 29 shows the L_2 norm of the error in velocity potential and the total number of active nodes in use as a function of grid density. It can be observed that:

- The error in the IBG still shows some oscillations during grid refinement, but with much smaller variance than in the submerged-cylinder case. These small oscillations are probably due to an even or odd number of grid nodes per wavelength. The random oscillations seen previously for the cylinder-case have disappeared. One reason for this is that the free-surface markers are all located along the vertical grid lines, which means that $x_{0c} = 0$ in Fig. 7a. As shown in Fig. 8a, this can result in smaller errors than if the markers are far away from the grid lines. Another reason is that the LE for the Dirichlet B.C. is more accurate than the LE for the Neumann B.C. at the markers in the cylinder case.
- The IBG method gives approximately 5th-order convergence, while the BFOG method has only 4th-order convergence. This can be understood from the local and global properties discussed in Section 2.2 and 3. The IBG method has 5th-order convergence for $E^D_{\varphi,GS}$, 6th-order convergence for $E^I_{\varphi,GS}$ and 4th-order convergence for $E^N_{\varphi,GS}$.

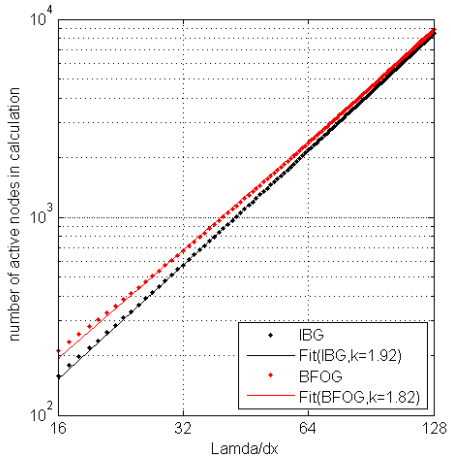
As an example, Fig. 30 provides the spatial distribution of $e_{\varphi,GS,rel}$ for grid density $\lambda/dx=32$. Fig. 30a shows that the error in the IBG method concentrates at the nodes involved in the free-surface Dirichlet B.C., which means that $E^D_{\varphi,GS}$ dominates $E_{\varphi,GS}$ so that its convergence is close to 5th order. For the BFOG method in Fig. 30b, the Dirichlet B.C. applies directly on nodes in the surface-fitted grid, so that $E^D_{\varphi,GS}$ is zero for these nodes. However, the free-surface-fitted grid is distorted, which will lead to a 4th-order convergence for the LE. Thus, in the free-surface-fitted grid $E^I_{\varphi,GS}$ has 2nd-order convergence, and $E^D_{\varphi,GS}$ has 5th-order convergence due to communication with the background grid. The background grid has 6th-order convergence for $E^I_{\varphi,GS}$ and 4th-order convergence for $E^N_{\varphi,GS}$ and $E^D_{\varphi,GS}$. The latter is due to communication with the boundary-fitted grid. The spatial error distribution indicates that the distortion of the boundary-fitted grid is the dominant error source. However, since there are only five layers of nodes in this grid, the overall error $E_{\varphi,GS}$ is approximately 4th order. It should also be noted that the distortion of the grid will be more severe if the wave steepness increases.

- The total number of active nodes involved in the calculations is similar for the IBG and BFOG methods for finer grids, while the BFOG method requires a slightly higher number of nodes when the grid is coarse. This is analogous to the submerged cylinder case.

Similarly as for the submerged cylinder case, enhanced accuracy is achievable for the wave tank by the treatments discussed in Section 4 or with the higher-order cells in Section 2.2.3.

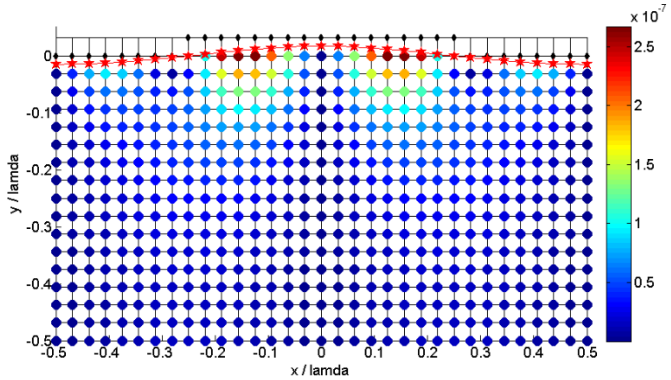


(a) $E_{\varphi,GS}$

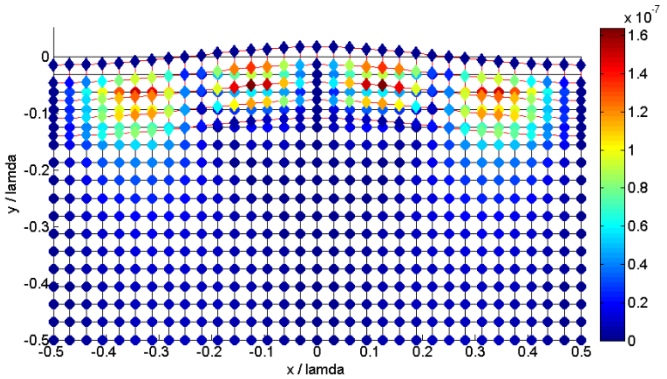


(b) Number of total active nodes

Fig. 29 Grid convergence of $E_{\varphi,GS}$ and total number of active nodes for numerical wave-tank



(a) Immersed boundary grid
(color filled circles for nodes in domain, small black filled diamonds for ghost nodes in use, and red stars for markers)



(b) Boundary-fitted overlapping grid
(color filled circles for background nodes, color filled diamonds for boundary-fitted nodes)

Fig. 30 Distribution of $e_{\varphi,GS,rel}$ ($\lambda/dx=32$, $t/T=0$)

After completing the analysis for the BVP problem above, we consider a case where the free surface evolves in time. For the IBG method the free surface is tracked by markers restricted to move vertically. The same approach is adopted for the BFOG method, i.e. the boundary-fitted nodes only move vertically along the vertical grid lines in the background grid. As a consequence, none of the methods is able to simulate overturning waves. It should be noted that this limitation can be removed if the markers or nodes are not fixed on the vertical grid line. The elevation and potential of the free surface are determined from the fully-nonlinear kinematic and dynamic free-surface conditions. These B.C.s give the time derivatives of the elevation and velocity potential on the markers or boundary-fitted nodes. The free-surface conditions are in the following written in semi-Lagrangian form. This notation is a modified version of the Mixed Eulerian-Lagrangian formulation first introduced by Ogilvie [25].

$$k_\eta = \frac{D\eta_{fs}}{Dt} = \frac{\partial\varphi}{\partial y} - \frac{\partial\varphi}{\partial x}\frac{\partial\eta}{\partial x}, \qquad on\ y = \eta\,; \tag{34}$$

$$k_\varphi = \frac{D\varphi_{fs}}{Dt} = -\frac{1}{2}\left(\left(\frac{\partial\varphi}{\partial x}\right)^2 + \left(\frac{\partial\varphi}{\partial y}\right)^2\right) - g\eta + \frac{\partial\varphi}{\partial y}\frac{d\eta_{fs}}{dt}, \qquad on\ y = \eta\,. \tag{35}$$

Here η_{fs} and φ_{fs} are the *y*-coordinate and velocity potential of a marker in the IBG method or of a grid node on the free surface in the BFOG method. A 4th-order explicit Runge-Kutta (RK4) scheme is used to integrate equations (34) and (35) in time. At every sub-step of the RK4 scheme, the vertical position and velocity potential of the markers or boundary-fitted grid nodes are updated, resulting in an intermediate free-surface state. Accordingly, the corresponding BVP must be solved at every sub-step in order to determine updated values of $\partial\varphi/\partial x$ and $\partial\varphi/\partial y$ to be used in equations (34) and (35). This means that for every time step, we have to solve the hydrodynamic BVP four times.

From equations (34) and (35) it is clear that the accuracy of k_η and k_φ depends on the accuracy at which $\partial\eta/\partial x$, $\partial\varphi/\partial x$ and $\partial\varphi/\partial y$ are evaluated after solving the BVP to determine the velocity potential in the entire fluid. Nevertheless, the fluid velocity components need only to be evaluated at the free-surface. This enables us to introduce some treatments to enhance the accuracy of these quantities with little additional computational cost.

- **Estimation of wave-slope** $\partial\eta/\partial x$

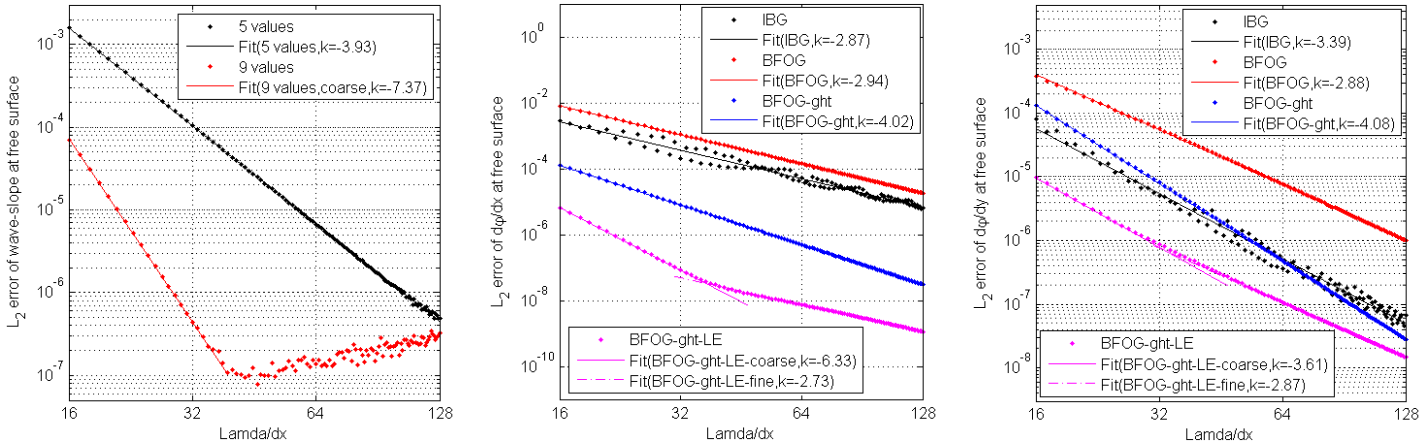
  In the present study, the wave-elevation is a single-valued function of the *x*-coordinate, and $\partial\eta/\partial x$ is determined at every marker (or node) through a FDM using the value of η at several adjacent markers (or nodes). We examine both a 4th-order FDM by using η-values at five consecutive markers (or nodes), and an 8th-order FDM by using η-values at nine consecutive markers (nodes). Fig. 31a shows the L_2 error of the estimated $\partial\eta/\partial x$ from the analytical wave elevation η_{true} at $t/T = 0$. This shows that the higher-order estimation significantly improves the accuracy for coarse grids. In this case, the error in $\partial\eta/\partial x$ converges between 7th and 8th order, which is much higher than the original 4th-order convergence when using lower-order estimation. The oscillatory behavior for finer grids is due to influence from round-off errors.

- **Estimation of velocities components** $\partial\varphi/\partial x$ **and** $\partial\varphi/\partial y$ **at the free surface**

  Once we have solved the BVP for the velocity potential, a straightforward way to estimate velocity components at the free surface is to use expression (8) directly with the same LE as already used to establish the equations for the global BVP. This may be feasible for $\partial\varphi/\partial y$, but can cause larger relative errors for $\partial\varphi/\partial x$. We explain this by using the cell in Fig. 7a as an example. The markers in the IBG method have $x_{0c} = 0$ and generally $|y_{0c}| \neq 0$ in Fig. 7a, while the free-surface nodes in the boundary-fitted grid have $x_{0c} = 0$ and $y_{0c} = dy$. These locations can have small relative errors in $\partial\varphi/\partial y$ as shown in Fig. 8c, but can also have (relatively speaking) large errors in $\partial\varphi/\partial x$ as shown in Fig. 8b. This is reflected by Fig. 31b and c, showing the L_2 error of the estimated velocity components at the free surface at $t/T = 0$. The notations 'IBG' and 'BFOG' refer to the IBG and BFOG methods. The errors convergence for the IBG method contains some small oscillations, with similar explanation as for the velocity potential in Fig.

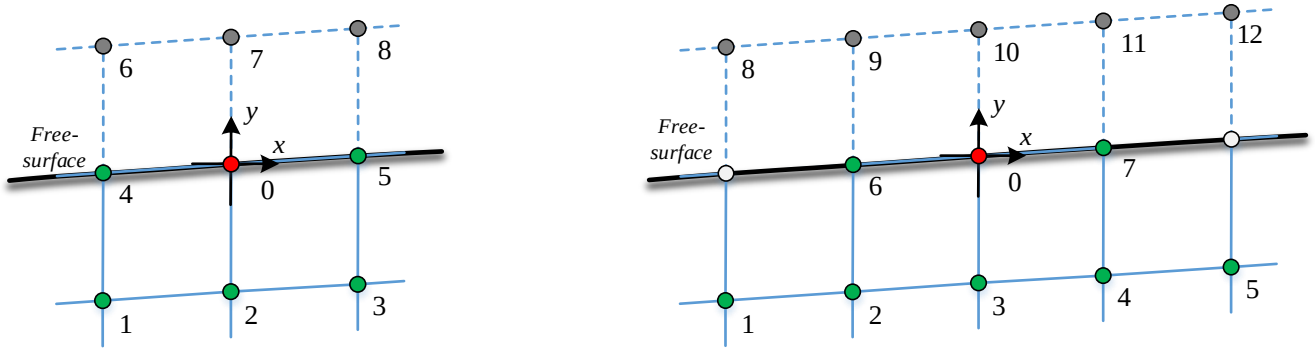
29a. For both the IBG method and the BFOG method, the errors in $\partial\varphi/\partial x$ are larger and converge slower than those for $\partial\varphi/\partial y$. Moreover, the IBG method has smaller errors than the BFOG method since the markers in this method generally have $|y_{0c}| < dy$.

One way to improve the accuracy in velocity estimations is to minimize $|y_{0c}|$ when evaluating the velocity components. This can be done easily in the BFOG method by simply adding a layer of ghost nodes above the free surface, meaning that the free surface becomes submerged in the enlarged grid where the BVP for the velocity potential is solved. For every ghost node, a new cell should be defined. Fig. 32a shows a scenario with a $N_{HP} = N_{node} = 8$ cell, where the LE (7) is applied as an additional equation for node 0 when solving the BVP for the velocity potential. Estimating velocity components for the same node and from the same cell, we can expect good accuracy since $|y_{0c}| = 0$. This is confirmed clearly in Fig. 31b and c, where results based on the cell in Fig. 32a are denoted 'BFOG-ght'. Both the magnitude and convergence rate of the errors are improved compared to the original BFOG method. In fact, a 4th-order order convergence is achieved for both $\partial\varphi/\partial x$ and $\partial\varphi/\partial y$. A further improvement can be achieved by using a $N_{HP} = N_{node} = 12$ cell to establish the LE for the velocity components as illustrated in Fig. 32b. According to Section 2.1, $\{h_j, j = 1 \sim 12\}$ should be selected. The LEs used in the BVP for the velocity potential remain unchanged, i.e. we use a higher-order cell purely to compute the velocities from the previously established velocity potential. The results from this are denoted 'BFOG-ght-LE' in Fig. 31b and c. Clearly, this results in the lowest L_2 errors for both velocity components. The convergence rate of the errors reduces for finer grids. This is mainly due to the limitations defined by the convergence of the velocity potential in Fig. 29a. The estimation of velocity components can be enhanced in the IBG method by similar strategies. In addition, the h-type or hp-type treatments in Section 4.1 are also applicable for both IBG and BFOG methods.



(a) wave-slope $\partial\eta/\partial x$ (b) horizontal velocity $\partial\varphi/\partial x$ (c) vertical velocity $\partial\varphi/\partial y$

Fig. 31 Grid convergence of wave-slope and velocities on the free-surface ($t/T = 0$)



(a) $N_{HP} = N_{node} = 8$ (b) $N_{HP} = N_{node} = 12$

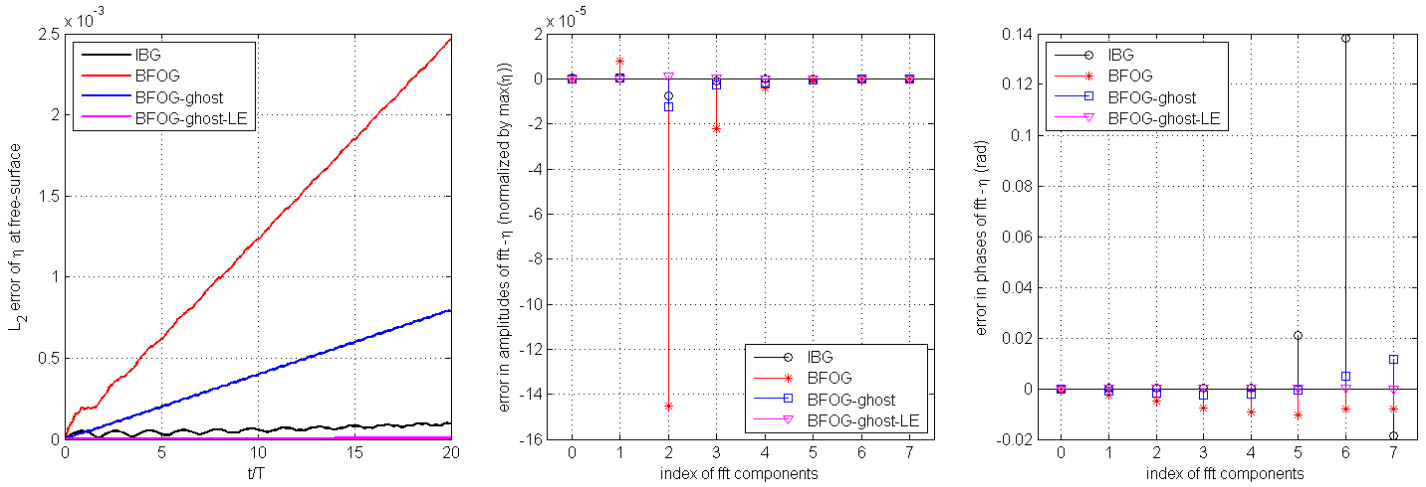
Fig. 32 Cells used for velocity estimation in the BFOG method with one layer of ghost nodes

(green filled circles are nodes in the fluid, dark filled circles are ghost nodes, red filled circles are nodes at which velocity components are estimated, and the white filled circles are nodes not used for the current LE)

The wave-slope and velocity components are estimated by the four methods outlined above, i.e. 'IBG', 'BFOG', 'BFOG-ght' and 'BFOG-ght-LE', for a case with $\lambda/dx = 32$, where the simulation has been continued for 20 wave periods T. The time-history of the L_2 error of η_{fs} is plotted in Fig. 33a. Here 'IBG' and 'BFOG' use a 4th-order FDM for $\partial\eta/\partial x$ without any additional treatment for velocity estimation, 'BFOG-ghost' uses a 4th-order FDM for $\partial\eta/\partial x$ and the cell in Fig. 32a for estimation of velocity components, and 'BFOG-ghost-LE' uses an 8th-order FDM for $\partial\eta/\partial x$ and the cell in Fig. 32b for estimation of velocity components. Detailed analysis is also carried out by taking the (spatial) discrete Fourier transform of η_{fs} at the last time step ($t/T = 20$). Corresponding errors in amplitude and phase for the first 8 Fourier-components are plotted in Fig. 33b and c. The contribution from higher-order components is negligible. Fig. 34 shows corresponding plots for φ_{fs}. The main observations are that:

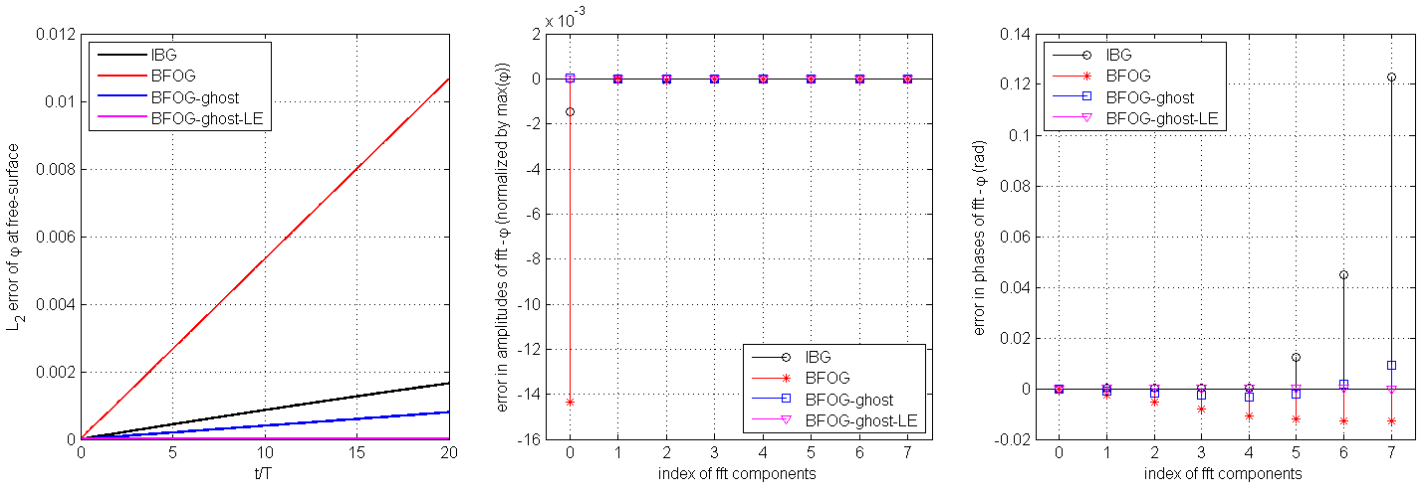
- The 'BFOG' method always has the largest errors among all the four methods, and shows the poorest conservation of amplitude and phase of η_{fs} and φ_{fs}. This is mainly due to low accuracy in the estimation of velocity components, which obviously is improved by adding one layer of ghost nodes as shown in results for 'BFOG-ghost'.
- The 'IBG' shows good accuracy and conservation properties for both η_{fs} and φ_{fs}. This is partially because the markers are moving only vertically along the vertical grid lines. The good behavior of the IBG method in this example cannot be guaranteed if the markers move with a fully Lagrangian strategy.
- The "BFOG-ghost-LE" method has the smallest errors, and conserves amplitudes and phases in the best manner.

In summary, the results in Fig. 34 show that there is a large potential to improve the accuracy significantly without increasing the computational effort much.



(a) history of L_2 error (b) error in amplitudes ($t/T = 20$) (c) error in phases ($t/T = 20$)

Fig. 33 Error in wave elevation ($\lambda/dx = 32$)



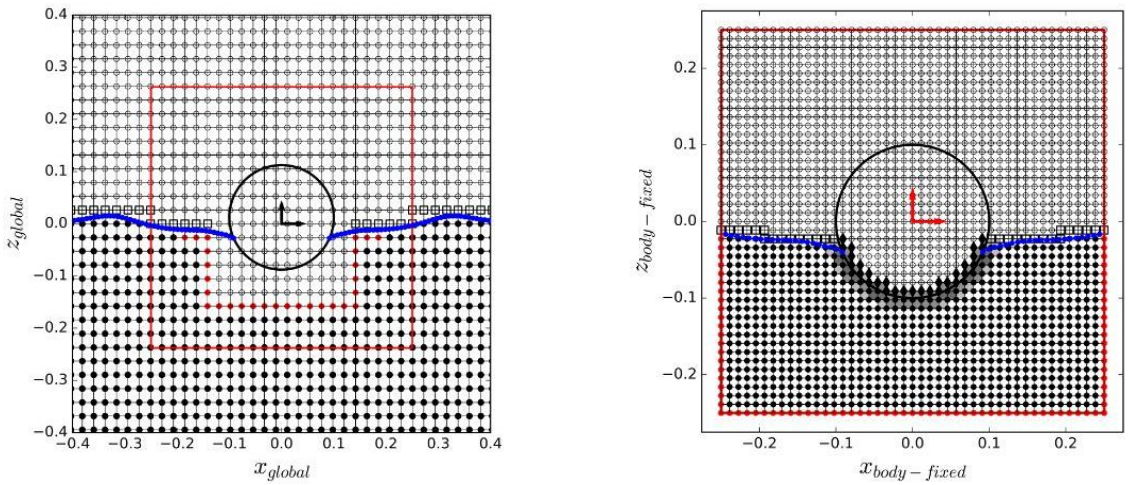
(a) history of L_2 error (b) error in amplitudes ($t/T = 20$) (c) error in phases ($t/T = 20$)

Fig. 34 Error in potential on the free-surface ($\lambda/dx = 32$)

4.2.3 Example 3: A heaving semi-submerged circular cylinder

The potential flow generated by a forced heaving circular cylinder semi-submerged in still water is studied in this sub-section. This problem involves nonlinear interaction between the rigid body surface and the deformable free surface. The case was previously studied numerically by Sun [23] applying a fully nonlinear BEM. Here we use the IBOG method and the point is to show the capability of the HPC method in solving complex practical problems.

The cylinder is placed in a numerical wave tank discretized by a coarse Earth-fixed background grid (BGG) and a denser overlapping body-fixed grid (BDG) surrounding the cylinder. Both BGG and BDG are structured, uniform grids with square cells, and the communication between them is carried out similarly as discussed in Section 4.2.1. Fig. 35 illustrates the grid systems used. In Fig. 35a, the BGG is seen from the Earth-fixed reference frame (with origin shown with the black coordinate axes) with the BDG indicated by the red square. In Fig. 35b, the BDG is seen from the body-fixed reference frame (with origin shown by red coordinate axes). In both grids, the free surface is indicated with blue circles, "standard" fluid nodes with black circles, free-surface ghost nodes with black open squares, grid-to-grid communication nodes with red circles, inactive nodes with open black circles and body-boundary ghost nodes with black diamonds.



(a) Background grid (b) Body-fixed grid

Fig. 35 The grid systems used for a semi-submerged heaving circular cylinder

The Neumann B.C. on the immersed body boundary is accounted for in a similar way as done in Section 4.2.1. The free surface is tracked by fully-Lagrangian markers that are evolved in time through a standard RK-4 scheme with temporal Lagrangian derivatives of the marker position (x_{fs}, y_{fs}) and velocity potential φ_{fs} expressed as

$$k_x = \frac{Dx_{fs}}{Dt} = \frac{\partial \varphi}{\partial x}, \quad k_y = \frac{Dy_{fs}}{Dt} = \frac{\partial \varphi}{\partial y}, \qquad on\ y = \eta; \tag{36}$$

$$k_\varphi = \frac{D\varphi_{fs}}{Dt} = \frac{1}{2}\left(\left(\frac{\partial \varphi}{\partial x}\right)^2 + \left(\frac{\partial \varphi}{\partial y}\right)^2\right) - g\eta, \qquad on\ y = \eta. \tag{37}$$

The Dirichlet B.C. for the velocity potential on the immersed free-surface is imposed in the same way as in Section 4.2.2. Since the markers are fully-Lagrangian, they will generally not coincide with vertical grid lines. According to the findings in the previous sections, the HPC solution can have better accuracy along the symmetry axes in a cell. In order to have this benefit, third-order B-splines are introduced to interpolate the values of the free-surface elevation and velocity potential so that the Dirichlet B.C. can be imposed along the vertical grid lines. Exceptions occur for wave-body intersection points, since these generally do not coincide with any grid lines. The reverse procedure applies when estimating spatial derivatives of the velocity potential at the free-surface, i.e. the derivative are evaluated at the same points in which the boundary conditions are imposed and thereafter interpolated back to the markers by third-order B-splines. Numerical beaches towards the tank end on both sides ensure that wave reflections from the tank walls are avoided.

At the intersection points between the free surface and body boundary, a double-node technique is deployed in order to satisfy both the free-surface Dirichlet B.C. and body-surface Neumann B.C. simultaneously. Since the body has a curved surface, the markers on the body-surface intersection points may separate during a time step. Thus, after each RK4 sub-step, these markers are projected normally back onto the body surface. When the heave oscillation frequency is sufficiently high, sharp angles may develop locally between the free surface and the body surface. In order to ensure numerical stability, a jet-cutting scheme is introduced when this angle becomes less than 5 degrees. The main steps of this scheme is to remove the marker on the body surface, project the second closest marker onto the body surface and redistribute the markers evenly in such a way that the number of markers remain unchanged. The jet-cutting is applied at the end of the RK4 and not during sub-steps. Redistribution of markers is performed after each complete RK4 step to ensure that the markers are evenly spaced along the global horizontal axis. A five-point spatial filter is then applied to the free-surface elevation and velocity potential to prevent instabilities from developing. During this process, the position of the wave-body intersection points are unchanged, but alteration of the velocity potential can occur.

As in [18], we consider a cylinder with radius $R = 0.1m$ subject to forced heave oscillations with non-dimensional amplitude $\varepsilon = z_a/R = 0.2$. Only the case with highest oscillation frequency corresponding to $\omega^2 R/g = 1.61$ is presented in the following. The wavelength of the outgoing wave can be estimated from the linear deep-water dispersion relation as $\lambda = 0.39m$. The length of the numerical wave tank is set to 10λ with the cylinder located in the center and the water depth is set to $15R$. The outer dimension of the BDG is $5R \times 5R$ with the cylinder located in the center. The grid size in the BGG is $\Delta x_{BGG} = \Delta y_{BGG} = 0.022m$ i.e. $\lambda/\Delta x_{BGG} \approx 18$ and in the BDG $\Delta x_{BDG} = \Delta y_{BDG} = 0.011m$ i.e. $R/\Delta x_{BDG} \approx 9$. This means that the ratio between the grid size in the BGG and the BDG is approximately equal to 2.0. Since strong free-surface curvatures may develop near the body surface intersection, a small time step of $\Delta t = 0.0025s$ is used, i.e. 200 time steps per oscillation period.

Fig. 36 compares snapshots of the free-surface elevation from the present simulation with those from Sun's BEM solution (obtained from personal communication) for one full heave cycle midway through the simulation. Apart for some deviations near the wave-body intersection point at $t/T = 5.6$ and $t/T = 5.8$, the results agree well. The deviations is possibly due to differences in the scheme used to account for the wave-body intersection point when sharp angles between the free surface and body surface occur. Fig. 37 shows the corresponding heave force on the cylinder, where the fluid pressure used in the force integration is found by solving a separate BVP for the acceleration potential. In the mid part of the time series the results from HPC and BEM match well, while there are some minor deviations for the initial and last part of the time series. Some small spikes can be observed in the global troughs, which is due to the jet-cutting scheme. Apart from this, the heave force is smooth and without unphysical oscillations. These results illustrate that the IBOG method can be used in a practical application with strong nonlinearities. The BFOG method introduced previously can also be applied, and similar results can be expected since it was found to be more stable than the IBOG method in Sections 4.2.1 and 4.2.2.

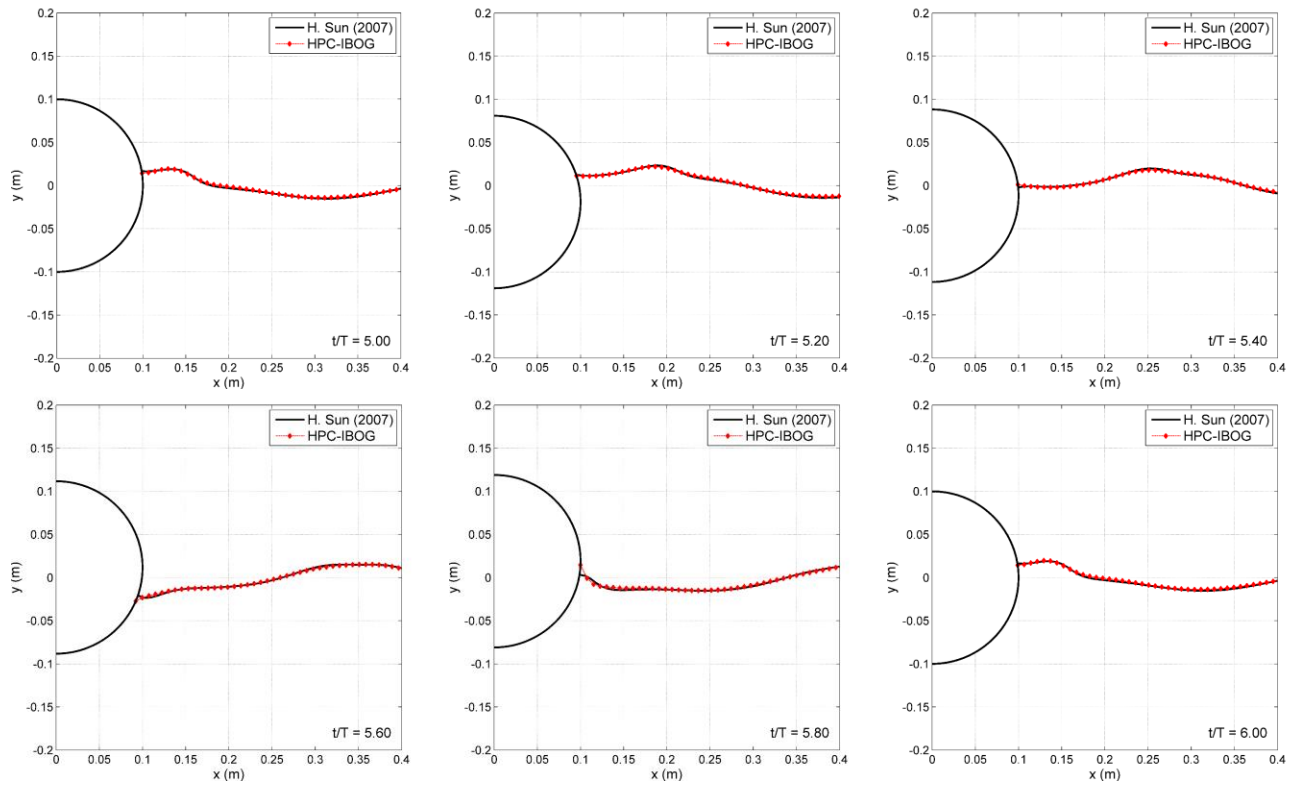


Fig. 36 Free-surface elevation at six different time steps

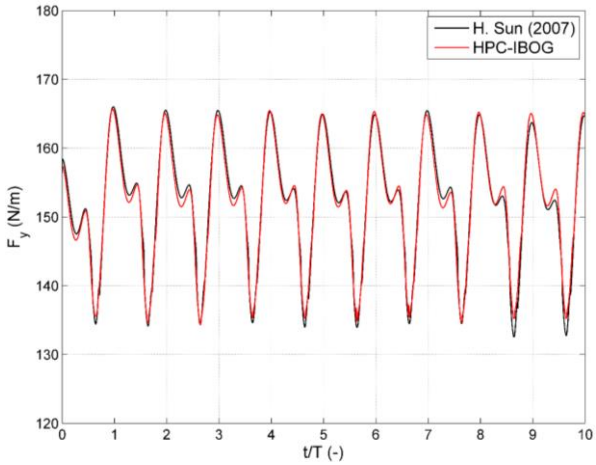


Fig. 37 Heave force acting on the semi-submerged circular cylinder

5. Conclusions and future studies

The present paper analyzes the properties of the HPC method, with focus on the LE and GS. The main purpose of this study is to determine how to harvest the full potential through a proper implementation of the HPC method. For HPC implementations in 2D, a set of general guidelines is suggested:

- The HPs used to form LEs should be chosen properly. For a general cell, the HPs can be chosen by using a method similar to [3]. In practice, one can achieve sufficient high accuracy with $N_{HP}=8\sim12$ and using the fact that the cells usually have symmetric node distribution. In this case, equations (9) give a more explicit and simpler rule to follow.
- The detailed discussions carried out regarding the accuracy in the LE and GS will help to guide towards a proper implementation of the method and an interpretation of the results. The grid-size convergence of the errors can be estimated from Section 2.2 and Section 3. The accuracy of the HPC method generally benefits from squared/symmetric cells, while severe stretching or distortion of the grid should be avoided. For overlapping grids

or immersed boundary methods, the interpolated nodes or markers should be located close to the cell center. For calculations involving Quadtree grid, the cells using 'parent' strategy usually give better accuracy. In order to achieve a smooth and efficient prediction of hydrodynamic forces, it is recommended to solve a BVP for the time derivative of velocity potential.

- For problems with large error contribution from boundary conditions, different treatments can be used to improve the accuracy with limited increase of computational effort. Among them, the hp-type method, using locally refined grid and high-order LE, is highly recommended.
- For complicated geometries and dynamic boundaries, three different methods have been introduced. The IBG and IBOG methods give larger oscillations in grid-size convergence studies, but can sometimes be more accurate than the BFOG method. Suggestions like ghost-node layer and high-order LEs for improving these three methods have been given in Section 4.

There also exist many topics for future studies. The efficiency of current methods regarding overlapping or immersed boundary grids can be further improved by avoiding change of matrix structure, so that preconditioning will not need to be done for every time step. The analyses in the paper are based on 2D HPC methods, and most of the conclusions can also be extended to 3D cases. The study for the properties of 3D HPC methods is currently in progress as a further study. Moreover, the analysis regarding accuracy in Section 2.2 and Section 3 can also be extended to the HPC-based Poisson solver in [9], and calculations have already shown that the original solver can be further improved. This work will be included in a future publication. At last, we also expect more implementations of the HPC method in the near future, which will help to further validate conclusions drawn in this paper.

Acknowledgement

The authors would like to acknowledge Dr. Claudio Lugni and Dr. Matteo Antuono for their useful suggestions. This work has been carried out at the Centre for Autonomous Marine Operations and Systems (AMOS). The Research Council of Norway is acknowledged as the main sponsor of AMOS. This work was supported by the Research Council of Norway through the Centre of Excellence funding scheme, Project number 223254-AMOS.

References

[1] Y.L. Shao and O.M. Faltinsen, Towards efficient full-nonlinear potential-flow solvers in marine hydrodynamics, Proceeding of the 31st International Conference on Ocean, Offshore and Arctic Engineering (OMAE), Rio De Janeiro, Brazil, 2012.

[2] Y.L. Shao and O.M. Faltinsen, Fully-nonlinear wave-current-body interaction analysis by a harmonic polynomial cell method, Journal of Offshore Mechanics and Arctic Engineering 136 (2014), 031301-031307

[3] Y.L. Shao and O.M. Faltinsen, A harmonic polynomial cell (HPC) method for 3D Laplace equation with application in marine hydrodynamics, Journal of Computational Physics 274 (2014), 312-332.

[4] H. Liang, O.M. Faltinsen and Y.L. Shao, Application of a 2D harmonic polynomial cell (HPC) method to singular flows and lifting problems, Applied Ocean Research 53 (2015), 75-90.

[5] A.G. Fredriksen, T. Kristiansen and O.M. Faltinsen, Experimental and numerical investigation of wave resonance in moonpools at low forward speed, Applied Ocean Research 47 (2014), 28-46.

[6] A.G. Fredriksen, T. Kristiansen and O.M. Faltinsen, Wave-induced response of a floating two-dimensional body with a moonpool, Philosophical Transactions of the Royal Society A 373 (2015).

[7] G. Colicchio, M. Greco and O.M. Faltinsen, Domain-decomposition strategy for marine applications with cavities entrapments, Journal of Fluids and Structures 27 (2011), 567-585.

[8] M. Greco, G. Colicchio and O.M. Faltinsen, A domain-decomposition strategy for a compressible multi-pha flow interacting with a structure, International Journal for Numerical Methods in Engineering 98 (2014), 840-858.

[9] A. Bardazzi, C. Lugni, M. Antuono, G. Graziani and O.M. Faltinsen, Generalized HPC method for the Poisson equation, Journal of Computational Physics 299 (2015), 630-648.

[10] I. N. Vekua, On the completeness of the system of harmonic polynomials in the space, Doklady AN USSR 90, 4 (1953), 495-498 (in Russian).

[11] F. Stern, R.V. Wilson, H.W. Coleman and E.G. Paterson, Comprehensive approach to verification and validation of CFD simulations-Part 1: methodology and procedures, Journal of Fluids Engineering 123, 4 (2001), 793-802.

[12] H. Sadat-Hosseini, P.C. Wu, P.M. Carrica, H. Kim, Y. Toda and F. Stern, CFD verification and validation of added resistance and motions of KVLCC2 with fixed and free surge in short and long head waves, Ocean Engineering 59 (2013), 240-273.

[13] M.M. Karim, B. Prasad and N. Rahman, Numerical simulation of free surface water wave for the flow around NACA 0015 hydrofoil using the volume of fluid (VOF) method, Ocean Engineering 78 (2014), 89-94.

[14] O.M. Faltinsen, Sea Loads on Ships and Offshore Structures, Cambridge University Press (1990).

[15] F.W. Hanssen, M. Greco, and Y.L. Shao, The harmonic polynomial cell method for moving bodies immersed in a Cartesian background grid, Proceeding of the 34st International Conference on Ocean, Offshore and Arctic Engineering (OMAE), St. John's, Newfoundland, Canada, 2015.

[16] M. Greco, A two-dimensional study of green-water loading, PhD thesis, Norwegian University of Science & Technology (NTNU), 2001.

[17] S. Yan and Q.W. Ma, Numerical simulation of fully nonlinear interaction between steep waves and 2D floating bodies using the QALE-FEM method, Journal of Computational Physics 221 (2007), 666-692

[18] P.J. Bandyk and R.F. Beck, The acceleration potential in fluid-body interaction problems, Journal of Engineering Mathematics 70 (2011), 147-163

[19] S. Ianniello, A.D. Mascio, A self-adaptive oriented particles Level-Set method for tracking interfaces, Journal of Computational Physics 229 (2010), 1353-1380.

[20] E. Aulisa, S. Manservisi, R. Scardovelli, A mixed markers and VOF method for the reconstruction and advection of interfaces in two-phase and free-boundary flows, Journal of Computational Physics 188 (2003), 611-639.

[21] A.P. Engsig-Karup, H.B. Bingham and O. Lindberg, An efficient flexible-order model for 3D nonlinear water waves, Journal of Computational Physics 228 (2009), 2100-2118

[22] M.M. Rienecker and J.D. Fenton, A Fourier approximation method for steady water waves, Journal of Fluid Mechanics 104 (1981), 119-137.

[23] H. Sun, A boundary element method applied to strongly nonlinear wave-body interaction problems, PhD thesis, Norwegian University of Science and Technology, 2007.

[24] O.M. Faltinsen and A.N. Timokha, Sloshing, Cambridge University Press (2009).

[25] T.F. Ogilvie, Nonlinear high-Froude-number free-surface problems, Journal of Engineering Mathematics 1, 3 (1967), 215-235.

[26] W. Cheney and D. Kincaid, Numerical Mathematics and Computing (7th edition), Cengage Learning (2012).

Appendix

Appendix A. Harmonic polynomials in 2D

Table 2. The HPs used in the paper

k	HPs		Even/Odd function	
	Original $h_k(x,y)$	Non-dimensional $h'_k(x,y)$	in respect of y axis	in respect of x axis

1	1	1	Even	Even
2	x	$(x)/\bar{r}$	Odd	Even
3	y	$(y)/\bar{r}$	Even	Odd
4	x^2-y^2	$(x^2-y^2)/\bar{r}^2$	Even	Even
5	$2xy$	$(2xy)/\bar{r}^2$	Odd	Odd
6	x^3-3xy^2	$(x^3-3xy^2)/\bar{r}^3$	Odd	Even
7	$3x^2y-y^3$	$(3x^2y-y^3)/\bar{r}^3$	Even	Odd
8	$x^4-6x^2y^2+y^4$	$(x^4-6x^2y^2+y^4)/\bar{r}^4$	Even	Even
9	$4x^3y-4xy^3$	$(4x^3y-4xy^3)/\bar{r}^4$	Odd	Odd
10	$x^5-10x^3y^2+5xy^4$	$(x^5-10x^3y^2+5xy^4)/\bar{r}^5$	Odd	Even
11	$5x^4y-10x^2y^3+y^5$	$(5x^4y-10x^2y^3+y^5)/\bar{r}^5$	Even	Odd
12	$x^6-15x^4y^2+15x^2y^4-y^6$	$(x^6-15x^4y^2+15x^2y^4-y^6)/\bar{r}^6$	Even	Even
13	$6x^5y-20x^3y^3+6xy^5$	$(6x^5y-20x^3y^3+6xy^5)/\bar{r}^6$	Odd	Odd
14	$x^7-21x^5y^2+35x^3y^4-7xy^6$	$(x^7-21x^5y^2+35x^3y^4-7xy^6)/\bar{r}^7$	Odd	Even
15	$7x^6y-35x^4y^3+21x^2y^5-y^7$	$(7x^6y-35x^4y^3+21x^2y^5-y^7)/\bar{r}^7$	Even	Odd
16	$x^8-28x^6y^2+70x^4y^4-28x^2y^6+y^8$	$(x^8-28x^6y^2+70x^4y^4-28x^2y^6+y^8)/\bar{r}^8$	Even	Even
17	$8x^7y-56x^5y^3+56x^3y^5-8xy^7$	$(8x^7y-56x^5y^3+56x^3y^5-8xy^7)/\bar{r}^8$	Odd	Odd

Appendix B. Proof for the rule to choose HPs for cells with symmetric node distributions

Firstly, a necessary and sufficient condition for achieving a non-singular $\mathbf{F}$ is derived. Then, (9) will be shown as a necessary but not sufficient condition. Here the cell has a symmetric node distribution with respect to the interpolated point and in addition $N_{HP}=N_{node}$. One of the coordinate axes should be set coinciding with the symmetry axis of the cell. The notations are the same as those in Section 2.1.4. The derivation starts by expressing the square matrix $\mathbf{F}$ as follows

$$\mathbf{F}=\left[\begin{array}{c:c}\mathbf{F}_{sym,even} & \mathbf{F}_{sym,odd}\\ \hdashline \mathbf{F}_{side1,even} & \mathbf{F}_{side1,odd}\\ \hdashline \mathbf{F}_{side2,even} & \mathbf{F}_{side2,odd}\end{array}\right]. \tag{38}$$

Here 'sym' refers to the N_{sym} rows for nodes along the symmetry axis, 'side1' refers to the $(N_{node}-N_{sym})/2$ rows for nodes on one side of the symmetry axis, and 'side2' refers to the $(N_{node}-N_{sym})/2$ rows for nodes on the other side of the symmetry axis. Moreover, 'even' refers to the $N_{HP,even}$ columns associated with even HPs and 'odd' refers to the $N_{HP,odd}$ columns associated with odd HPs. According to the definition of even and odd functions, $\mathbf{F}_{sym,odd}=0$ and the rows can be arranged so that $\mathbf{F}_{side1,even}=\mathbf{F}_{side2,even}$ and $\mathbf{F}_{side1,odd}=-\mathbf{F}_{side2,odd}$. These properties inspire us to further simplify (38) by a transformation matrix $\mathbf{T}$

$$\mathbf{F}=\mathbf{T}\mathbf{F}', \tag{39}$$

$$\mathbf{T}=\left[\begin{array}{c|c|c} \mathbf{I}_{N_{sym}} & 0 & 0 \\ \hline 0 & \mathbf{I}_{\left(N_{node}-N_{sym}\right)/2} & \mathbf{I}_{\left(N_{node}-N_{sym}\right)/2} \\ \hline 0 & \mathbf{I}_{\left(N_{node}-N_{sym}\right)/2} & -\mathbf{I}_{\left(N_{node}-N_{sym}\right)/2} \end{array}\right],\quad \mathbf{F}'=\left[\begin{array}{c|c} \mathbf{F}_{sym,even} & 0 \\ \hline \mathbf{F}_{side1,even} & 0 \\ \hline 0 & \mathbf{F}_{side1,odd} \end{array}\right]=\left[\begin{array}{c|c} \mathbf{F}_{even} & 0 \\ \hline 0 & \mathbf{F}_{odd} \end{array}\right],$$

where $\mathbf{I}$ is an unit matrix, $\mathbf{F}_{even}=\left[\begin{array}{c} \mathbf{F}_{sym,even} \\ \hline \mathbf{F}_{side1,even} \end{array}\right]$ and $\mathbf{F}_{odd}=\mathbf{F}_{side1,odd}$. Moreover, $\mathbf{F}_{even}$ has dimensions $\left(\left(N_{node}+N_{sym}\right)/2\right)\times N_{HP,even}$ and $\mathbf{F}_{odd}$ has dimensions $\left(\left(N_{node}-N_{sym}\right)/2\right)\times N_{HP,odd}$. As shown previously for the $N_{BF}=N_{node}$ case, $\mathbf{F}$ should be invertible and this is equivalent to $\det\left(\mathbf{F}\right)\neq 0$. Since $\det\left(\mathbf{F}\right)=\det\left(\mathbf{T}\right)\cdot\det\left(\mathbf{F}'\right)$ and $\det\left(\mathbf{T}\right)\neq 0$, the analysis will be focused on whether $\mathbf{F}'$ is singular or not. This can be determined from the rank of the matrix:

$$\operatorname{rank}\left(\mathbf{F}'\right)=\operatorname{rank}\left(\mathbf{F}_{even}\right)+\operatorname{rank}\left(\mathbf{F}_{odd}\right)\ . \tag{40}$$

The matrix $\mathbf{F}'$ is invertible if and only if $\operatorname{rank}\left(\mathbf{F}'\right)=N_{node}$. By using the following relations

$$\begin{aligned} &\operatorname{rank}\left(\mathbf{F}_{even}\right)\le\min\left\{\left(N_{node}+N_{sym}\right)/2,N_{HP,even}\right\} \\ &\operatorname{rank}\left(\mathbf{F}_{odd}\right)\le\min\left\{\left(N_{node}-N_{sym}\right)/2,N_{HP,odd}\right\} \\ &\left(N_{node}+N_{sym}\right)/2+\left(N_{node}-N_{sym}\right)/2=N_{node}=N_{HP}=N_{HP,even}+N_{HP,odd} \end{aligned},$$

it is easy to show that the necessary and sufficient condition for $\mathbf{F}$ to be invertible is

$$\begin{cases} \operatorname{rank}\left(\mathbf{F}_{even}\right)=\left(N_{node}+N_{sym}\right)/2=N_{HP,even} \\ \operatorname{rank}\left(\mathbf{F}_{odd}\right)=\left(N_{node}-N_{sym}\right)/2=N_{HP,odd} \end{cases}. \tag{41}$$

Comparing (9) with (41), it can be seen that the former is a necessary but not sufficient condition.

Appendix C. Dependence of error in global solution on the condition number

A brief proof for (22) will be shown in this section using the same notations as in Section 3. Note that only the error induced by discretization of the governing equation is included here, and the errors due to boundary conditions are not considered. The following formulations are based on [26]. According to (16)

$$\mathbf{A}\boldsymbol{\varphi}_{calc,GS}=\boldsymbol{b}_{\phi}\ \ \Rightarrow\ \ \left\|\mathbf{A}\right\|_2\left\|\boldsymbol{\varphi}_{calc,GS}\right\|_2\ge\left\|\boldsymbol{b}_{\phi}\right\|_2\ \ \Rightarrow\ \ \left\|\boldsymbol{\varphi}_{calc,GS}\right\|_2\ge\left(\left\|\mathbf{A}\right\|_2\right)^{-1}\left\|\boldsymbol{b}_{\phi}\right\|_2\ . \tag{42}$$

Moreover, according to (21),

$$\mathbf{A}\boldsymbol{e}^{I}_{\varphi,GS}=\boldsymbol{b}^{I}_{e}\ \ \Rightarrow\ \ \boldsymbol{e}^{I}_{\varphi,GS}=\mathbf{A}^{-1}\boldsymbol{b}^{I}_{e}\ \ \Rightarrow\ \ \left\|\boldsymbol{e}^{I}_{\varphi,GS}\right\|_2\le\left\|\mathbf{A}^{-1}\right\|_2\left\|\boldsymbol{b}^{I}_{e}\right\|_2. \tag{43}$$

Dividing (43) by (42), and introducing the condition number as $\operatorname{cond}\left(\mathbf{A}\right)=\left\|\mathbf{A}^{-1}\right\|_2\left\|\mathbf{A}\right\|_2$, we have

$$\frac{\left\|\boldsymbol{e}^{I}_{\varphi,GS}\right\|_2}{\left\|\boldsymbol{\varphi}_{calc,GS}\right\|_2}\le\operatorname{cond}\left(\mathbf{A}\right)\cdot\frac{\left\|\boldsymbol{b}^{I}_{e}\right\|_2}{\left\|\boldsymbol{b}_{\phi}\right\|_2}. \tag{44}$$

It is possible to show that (44) is an approximation of the error $E^{I}_{\varphi,GS}$ when the solution error $\boldsymbol{e}^{I}_{\varphi,GS}$ is sufficiently small. In particular, we have $\boldsymbol{\varphi}_{calc,GS}=\boldsymbol{\varphi}_{true,GS}+\boldsymbol{e}^{I}_{\varphi,GS}$. The error can be positive or negative but in both cases $\left\|\boldsymbol{\varphi}_{calc,GS}\right\|_2\le\left\|\boldsymbol{\varphi}_{true,GS}\right\|_2+\left\|\boldsymbol{e}^{I}_{\varphi,GS}\right\|_2$, therefore if $\left\|\boldsymbol{e}^{I}_{\varphi,GS}\right\|_2<<\left\|\boldsymbol{\varphi}_{true,GS}\right\|_2$ we have that $\left\|\boldsymbol{\varphi}_{calc,GS}\right\|_2\approx\left\|\boldsymbol{\varphi}_{true,GS}\right\|_2$. Thus

$$E^{I}_{\varphi,GS} = \frac{\left\| \boldsymbol{e}^{I}_{\varphi,GS} \right\|_2}{\left\| \boldsymbol{\varphi}_{true,GS} \right\|_2} \approx \frac{\left\| \boldsymbol{e}^{I}_{\varphi,GS} \right\|_2}{\left\| \boldsymbol{\varphi}_{calc,GS} \right\|_2} . \tag{45}$$

Thereby (22) is proved by combining (44) and (45).