 CMES, vol.9, no.2, pp.193-209, 2005

MLPG Method Based on Rankine Source Solution for Simulating Nonlinear Water Waves

Q.W. Ma [1]

Abstract: Recently, the MLPG (Meshless Local Petrov-Galerkin Method) method has been successfully extended to simulating nonlinear water waves [Ma, (2005)]. In that paper, the author employed the Heaviside step function as the test function to formulate the weak form over local sub-domains, acquiring an expression in terms of pressure gradient. In this paper, the solution for Rankine sources is taken as the test function and the local weak form is expressed in term of pressure rather than pressure gradient. Apart from not including pressure gradient, velocity gradient is also eliminated from the weak form. In addition, a semi-analytical technique is developed to estimate the domain integral involved in this method, rendering it unnecessary to evaluate velocities at a quite large number of points. These features potentially make the numerical discretisation of the governing equations relatively easier and more efficient. The method is validated by simulating various water waves generated by a wavemaker and by motions of a tank. Good agreement of results with those from publications is achieved.

keyword: Water waves; Meshless Local Petrov-Galerkin method (MLPG); MLPG_R; Free surface

1 Introduction

Nonlinear water waves are of great concern to offshore and costal engineering but are difficult to deal with. Simulating them has received numerous studies, for which mesh-based methods, such as finite element (FEM), finite volume and finite difference methods, are widely used. These methods have provided many useful and satisfactory results. However, their successes largely rely on good quality meshes. The construction of such meshes, particularly unstructured, is usually a difficult and time-consuming task because it must be ensured that aspect ratios of all elements are not very large and not very small and connectivity between nodes and elements must be carefully and accurately found and recorded. More problematically, elements can frequently become over-distorted during simulation of water wave evolutions. The over-distorted elements may be amended by remeshing. However remeshing can be as expensive as the generation of original meshes and may take a major proportion of computational time considering that this has to be done every time step if unstructured mesh is necessarily used. In order to tackle the difficult, the research group of the author of this paper devised a new method called quasi arbitrary Lagrangian-Eulerian finite element method (QALE-FEM) [Ma and Yan (2005)]. The main difference of the QALE-FEM from the conventional FEMs is that the complex unstructured mesh is generated only once at the beginning and is moved by using a spring analogy method at all other time steps in order to conform to the motion of the free surface. This feature allows one to use an unstructured mesh with any degree of complexity without the need of regenerating it at every time step and so the difficulty with regenerating meshes is bypassed. However, the QALE-FEM is so far suitable only for potential flow problems. Although the over-distortion problems may not arise if using fixed meshes, numerical diffusions due to advection terms may become severe and the motion of a floating body is not easy to cope with in such cases.

A new class of methods has recently been developed, which do not pose the problems associated with over-distortion. These methods do not need use of any mesh to discretise computational domains and to construct approximate solutions. Alternatively, they are only based on randomly-ordered and -distributed nodes, implying that the problems associated with the shape and connectivity of elements and regeneration of mesh in mesh-based methods vanish automatically in these meshless (or particle) methods. Many Meshless (or particle) methods have been reported in the literature, such as element free Galerkin method [Belytschko, Lu and

[1] School of Engineering and Mathematical Sciences, City University, Northampton Square, London, UK, EC1V 0HB

Gu (1994)], diffusion element method [Nayroles, Nayroles, Touzot and Villon (1992)], reproducing kernel particle method [Liu, Chen, Jun, Chen, Belytschko, Pan, Uras, and Chang (1996)], smoothed particle hydrodynamics method [Monaghan (1994)], particle finite element method [Onate, Idelsohn, Pin and Aubry (2004)] and so on. Among them, the smoothed particle hydrodynamics (SPH) method has been most widely used to simulate water wave problems, see, e.g., Monaghan (1994), Dalrymplem and Knio (2001) and Lo and Shao (2002).

More recently, another meshless method, called Meshless Local Petrove-Galerkin (MLPG) method, has been developed in Atluri and Zhu (1998) and Atluri and Shen (2002). This method is based on a local weak form over local sub-domains (circles for two dimensional problems and spheres for three dimensional ones). It offers great flexibility. Many other meshless methods can be considered to be special cases of the MLPG method. The success of the MLPG method has been reported in solving fracture mechanics problems [Batra and Ching (2002)], beam and plate bending problems [Atluri and Zhu (2000)], three dimensional elasto-static and -dynamic problems [Han and Atluri (2004a,b)] and some fluid dynamic problems such as steady flow around a cylinder [Atluri and Zhu (1998)], steady convection and diffusion flow [Lin and Atluri (2000)] in one and two dimensions and lid-driven cavity flow in a two dimensional box [Lin and Atluri (2001)].

In Ma (2005), the MLPG method was extended to simulating nonlinear water waves and produced some encouraging results. In that paper, the simple Heaviside step function was adopted as the test function to formulate the weak form over local sub-domains, resulting in one in terms of pressure gradient.

In this paper, the MLPG method is still used but the solution for Rankine sources, which is very popular for solving nonlinear water wave problems by using conventional boundary element methods, will be taken as the test function. Although this test function seems to be more intricate than the Heaviside step function, the resulting weak form does not contain gradients or derivatives of unknown functions, which potentially make numerical discretisation of the governing equations relatively easier and more efficient. For the ease of description, the MLPG method based on the Rankine source solution is named as MLPG_R method in this paper. Similar approach has been suggested, e.g., in Zhu, Zhang and Atluri (1998) and Sellountos and Polyzos (2003), for the LBIE method and applied to some solid and fluid problems but not to nonlinear water wave problems. The difference of this work from other publications using the LBIE method is that the approach is extended to dealing with nonlinear water waves and a special local weak form particularly suitable for this kind of problems is derived.

2 Governing Equation and Numerical Procedure

The flow of incompressible and nonviscous fluids is considered, which is governed by the following equations and conditions:

$$\nabla \cdot \vec{u} = 0 \tag{1a}$$

$$\frac{D\vec{u}}{Dt} = -\frac{1}{\rho}\nabla p + \vec{g} \tag{1b}$$

$$\frac{D\vec{x}}{Dt} = \vec{u} \text{ and } p = p_{atm} \text{ on the free surface} \tag{2a}$$

$$\vec{u}.\vec{n} = \vec{U}.\vec{n} \text{ and } \vec{n}.\nabla p = \rho\left(\vec{n}\cdot\vec{g} - \vec{n}\cdot\dot{\vec{U}}\right) \text{ on solid boundaries} \tag{2b}$$

where, ρ is the density of fluids, $\vec{u}$ the velocity of fluids, $\vec{U}$ the velocity of solid boundaries, $\vec{g}$ the gravitational acceleration, p the pressure and p_{atm} the atmospheric pressure.

It is well known that the nonviscous flow can be formulated by a velocity potential. Nevertheless, the potential formulation is not adopted here, because our research efforts will not be restricted to nonviscous flow in future.

The above equations will be solved using a time-step marching procedure. This starts from a particular instant when the velocity and the geometry of fluid flow are known and then evolves to next time step, at which all physical quantities are updated by solving the governing equations. During each time step, the problem is formulated using a well known time-split procedure. This formulation contains three sub-steps:

1. Evaluating intermediate velocities and positions.

   At start of a new time step, intermediate velocities and positions are first evaluated by

   $$\vec{u}^{(*)} = \vec{u}^{(n)} + \vec{g}\Delta t \tag{3}$$

   $$\vec{r}^{(*)} = \vec{r}^{(n)} + \vec{u}^{(*)}\Delta t \tag{4}$$

where $\vec{r}$ is the position vector of a point; Δt is the time step; and the superscripts (n) indicate the quantities at time$t = t_n$. On this basis, the velocities at time $t = t_{n+1} = t_n + \Delta t$ can be expressed by

$$\vec{u}^{(n+1)} = \vec{u}^{(*)} + \vec{u}^{(**)}. \tag{5}$$

2. Estimating the pressure

   Integrating Eq. (1b) over the time interval (t_n, t_{n+1}) results in

$$\vec{u}^{(n+1)} = \vec{u}^{(n)} + \vec{g}\Delta t - \int_{t_n}^{t_{n+1}} \left(\frac{1}{\rho}\nabla p\right) dt. \tag{6}$$

   The integration with respect to time can be approximated by well-known methods, such as explicit, implicit or semi-implicit methods, i.e.

$$\int_{t_n}^{t_{n+1}} \left(\frac{1}{\rho}\nabla p\right) dt = \Delta t \left[\frac{\theta}{\rho}\nabla p^{(n+1)} + \frac{1-\theta}{\rho}\nabla p^{(n)}\right] \tag{7}$$

   with θ=0 for explicit, θ=1 for implicit and θ=1/2 for semi-implicit methods. Substituting Eq. (7) into Eq. (6) results in:

$$\begin{aligned}\vec{u}^{(n+1)} &= \vec{u}^{(n)} + \vec{g}\Delta t - \Delta t \left[\frac{\theta}{\rho}\nabla p^{(n+1)} + \frac{1-\theta}{\rho}\nabla p^{(n)}\right] \\ &= \vec{u}^{(*)} - \Delta t \left[\frac{\theta}{\rho}\nabla p^{(n+1)} + \frac{1-\theta}{\rho}\nabla p^{(n)}\right].\end{aligned} \tag{8}$$

   From Eq. (5) it follows that

$$\vec{u}^{(**)} = -\frac{\Delta t}{\rho}\left[\theta\nabla p^{(n+1)} + (1-\theta)\nabla p^{(n)}\right]. \tag{9}$$

   The velocity in Eq. (5) or (8) must also satisfy the continuity equation (1a), yielding

$$\nabla^2 p^{(n+1)} = \frac{\rho}{\theta\Delta t}\nabla\cdot\vec{u}^{(*)} - \left(\frac{1}{\theta} - 1\right)\nabla^2 p^{(n)}. \tag{10}$$

   It is clear that this formulation is not suitable for θ=0. In fact, the explicit method corresponding to this value of θ is not stable and possesses low accuracy. Thus, it is suggested that the value of θ is always chosen in the range$0 < \theta \leq 1$. In this paper, the fully implicit method (θ=1) is employed, leading to

$$\vec{u}^{(n+1)} = \vec{u}^{(*)} - \frac{\Delta t}{\rho}\nabla p^{(n+1)} \tag{11}$$

   and

$$\nabla^2 p^{(n+1)} = \frac{\rho}{\Delta t}\nabla\cdot\vec{u}^{(*)}. \tag{12}$$

   Eq. (10) or Eq. (12) governs the pressure at the new time level, from which the solution for the pressure at the new time level can be found.

3. omputing the velocity and the position at time$t = t_{n+1}$.

   After the solution for $p^{(n+1)}$ is found, the velocities can be estimated using Eq. (11) and the positions of fluids can be updated by numerically integrating the velocities.

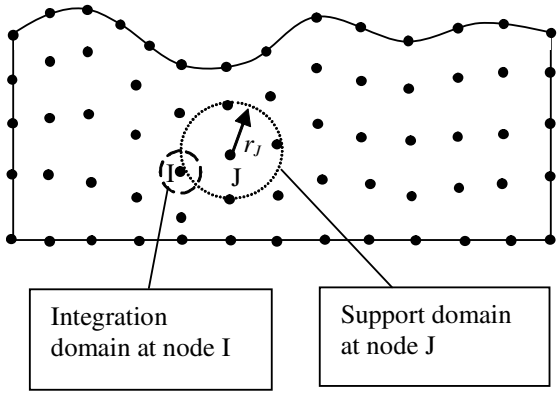


Figure 1 : Illustration of nodes, integration domain and support domain

The time split procedure was initially introduced by Chorin (1968) for incompressible flow. It has been extended to deal with various problems using a finite element method as described in the book of Zienkiewicz and Taylor (2000). The procedure has also been employed by a number of authors who use other meshless or particle methods, e.g., Lo and Shao (2002), Koshizuka, Nobe, Oka (1998) and Idelsohn, Storti, Onate (2001).

Although the above formulation is employed for the non-viscous flow it may be extended to viscous flow by adding viscous stress terms to Eq. (3). In that case,

however, iterations may be necessary since the viscous stresses are estimated using the velocities at previous steps, which would require to be improved for dealing with complicated phenomena in viscous flows.

3 MLPG_R Formulations

The key task in the above formulation is to find the solution for the pressure by solving Eq. (12). Various methods, such as finite element and finite different methods may be used but in this paper the MLPG_R method will be employed, as indicated above. This method is based on a set of nodes, as illustrated in Fig. 1, which discretise the fluid domain. Some of these nodes are located on boundaries and others lie inside the fluid domain. Formulation around inner nodes is first discussed and formulation around boundary nodes is described in Section 5. At each of the inner nodes, a sub-domain is specified, which is a circle for two-dimensional (2D) and a sphere for three-dimensional (3D) cases. Eq. (12), after multiplying by an arbitrary test function ϕ, is integrated over the sub-domain, leading to

$$\int_{\Omega_I} \left[\nabla^2 p - \frac{\rho}{\Delta t}\nabla\cdot\vec{u}^{(*)}\right]\varphi d\Omega = 0 \tag{13}$$

where Ω_I is the area or volume of the sub-domain centred at node I. There are many options for the test function [Atluri and Shen (2002)]. In Ma (2005), the Heaviside step function was used. In the MLPG_R method, the solution for Rankine sources is taken as the test function. This test function φ is made to satisfy that$\nabla^2\varphi = 0$, in Ω_I except for its centre and$\varphi = 0$, on $\partial\Omega_I$, the boundary of Ω_I. The solution satisfying these equation and conditions is well known and can be expressed as

$$\varphi = \frac{1}{4\pi}\begin{cases} 2\ln(r/R_I) & \text{for a two dimensional case} \\ (1 - R_I/r) & \text{for a three dimensional case} \end{cases} \tag{14}$$

where r is the distance between a concerned point and the centre of Ω_I and R_I the radius of Ω_I.

In Eq. (13), the second order derivative of unknown pressure and the gradient of the intermediate velocity are included. Numerical calculation of the derivative and gradient requires not only much computational time but also degrades the accuracy. In order to obtain a better form, Eq. (13) is changed, by adding a zero term $p\nabla^2\varphi$ and applying the Gauss's theorem, into

$$\begin{aligned}&\int_{\partial\Omega_I+\partial\varepsilon} [\vec{n}\cdot(\varphi\nabla p) - \vec{n}\cdot(p\nabla\varphi)]\, dS \\ &= \int_{\partial\Omega_I+\partial\varepsilon} \frac{\rho}{\Delta t}\vec{n}\cdot\left(\varphi\vec{u}^{(*)}\right) d\Omega - \int_{\Omega_I} \frac{\rho}{\Delta t}\vec{u}^{(*)}\cdot\nabla\varphi\, d\Omega\end{aligned} \tag{15}$$

where $\partial\varepsilon$ is a small surface surrounding the centre of Ω_I, which is a circle in 2D cases and a spherical surface in 3D cases with a radius of ε. The reason for adding $\partial\varepsilon$ is that the test function ϕ in Eq. (14) becomes infinite at r = 0 and so the Gauss's theorem can not be used otherwise. One can easily prove that taking $\varepsilon \to 0$results in

$$\int_{\partial\Omega_I+\partial\varepsilon} [\vec{n}\cdot(\varphi\nabla p)]\, dS = 0,$$

$$\int_{\partial\varepsilon} [\vec{n}\cdot(p\nabla\varphi)]\, dS = -p \quad ,$$

and

$$\int_{\partial\Omega_I+\partial\varepsilon} n\cdot\varphi\vec{u}^{(*)} dS = 0.$$

As a result, Eq. (13) eventually becomes

$$\int_{\partial\Omega_I} \vec{n}\cdot(p\nabla\varphi)\, dS - p = \int_{\Omega_I} \frac{\rho}{\Delta t}\vec{u}^{(*)}\cdot\nabla\varphi\, d\Omega. \tag{16}$$

Remarkable features of Eq. (16) are worthy to be discussed. This equation is similar to Eq. (9) in Ma (2005) in that only boundary integral on the left hand side is involved but is distinct from the latter in that the pressure, rather than pressure gradient, is dealt with here. This is of great benefit because estimating pressure is much easier than estimating pressure gradient. In addition, although the left hand side of Eq. (16) is in a similar form to that of Eq. (9) in Zhu, Zhang and Atluri (1998), the right hand side is different from the corresponding term in that paper. It is this difference that allows the term related to the intermediate velocity derivatives to be eliminated from the weak form. The improvement in these two aspects compared with previous publications makes the evaluation of the integrals and therefore the whole procedure more efficient in the MLPG_R method.

4 Weight and Shape Functions

The unknown function p needs to be approximated by a set of discretised variables. Generally, the approximation may be written as

$$p(\vec{x}) \approx \sum_{j=1}^{N} \Phi_j(\vec{x})\,\hat{p}_j \tag{17}$$

where N is the number of nodes that affect the pressure at point $\vec{x}$; $\hat{p}_j$ are nodal variables but not necessarily equal to the nodal values of $p(\vec{x})$; and $\Phi_j(\vec{x})$ interpolation functions called shape functions as they play a similar role to shape functions in finite element methods. However in finite element methods, the shape functions are formulated by assuming that the unknown function is described by an explicit function depending on the size of elements and connectivity between nodes and elements. In meshless methods, there are no elements and no connectivity between nodes stored. Therefore, the formulation of the shape functions here is entirely different from that for finite element methods. In general, a local approximation to the unknown function is assumed in meshless methods, which is expressed in terms of unknown variables corresponding to some randomly located nodes nearby. This local approximation may be formulated in a variety of ways. One of them is to use a moving least-square (MLS) method (Atluri and Shen, 2002), which is adopted in our work. With this method, the shape function is given by

$$\begin{aligned}\Phi_J(\vec{x}) &= \sum_{m=1}^{M} \psi_m(\vec{x}) \left[\mathbf{A}^{-1}\left(\vec{x}\right)\mathbf{B}(\vec{x})\right]_{mJ} \\ &= \psi^T(\vec{x})\mathbf{A}^{-1}\left(\vec{x}\right)\mathbf{B}_J(\vec{x})\end{aligned} \tag{18}$$

with the base function being $\psi^T(\vec{x}) = [\psi_1, \psi_2, \psi_3] = [1, x, y]$(M=3) for 2D
and $\psi^T(\vec{x}) = [\psi_1, \psi_2, \psi_3, \psi_4] = [1, x, y, z]$(M=4) for 3D;
and with the matrixes $\mathbf{B}(\vec{x})$ and $\mathbf{A}(\vec{x})$ being defined as

$$\begin{aligned}\mathbf{B}(\vec{x}) &= \Psi^T \mathbf{W}(\vec{x}) \\ &= [w_1(\vec{x}-\vec{x}_1)\,\psi(\vec{x}_1), w_2(\vec{x}-\vec{x}_2)\,\psi(\vec{x}_2), \cdots]\end{aligned} \tag{19}$$

and

$$\mathbf{A}(\vec{x}) = \Psi^T \mathbf{W}(\vec{x})\,\Psi = \mathbf{B}(\vec{x})\,\Psi \tag{20}$$

where $\mathbf{W}(\vec{x})$ and Ψ are, respectively, expressed by

$$\mathbf{W}(\vec{x}) = \begin{bmatrix} w_1(\vec{x}-\vec{x}_J) & 0 \cdots & 0 \\ 0 & & \\ \cdots & & \\ 0 & & w_N(\vec{x}-\vec{x}_J) \end{bmatrix} \tag{21a}$$

and

$$\Psi^T = [\psi(\vec{x}_1), \psi(\vec{x}_2), \cdots, \psi(\vec{x}_N)] \tag{21b}$$

which shows each column of the matrix Ψ^T is the value of the base function ψ at a particular point. The weight function $w_J(\vec{x}-\vec{x}_J)$ may be chosen to be a spline function given by

$$w_J(\vec{x}-\vec{x}_J) = \begin{cases} 1-6\left(\frac{d_J}{r_J}\right)^2 + 8\left(\frac{d_J}{r_J}\right)^3 - 3\left(\frac{d_J}{r_J}\right)^4 \\ \qquad 0 \le \frac{d_J}{r_J} \le 1 \\ 0 \qquad\qquad \frac{d_J}{r_J} > 1 \end{cases} \tag{22}$$

where r_J is the size of support domain of the weight function (see Fig. 1) and $d_J = |\vec{x}-\vec{x}_J|$ the distance between the node J and the point $\vec{x}$. In order to ensure the shape function is well defined, the number of nodes (N) affecting the concerned point must be larger than M, i.e. $N \geq 3$ in 2D and $N \geq 4$ in 3D cases.

After the pressure is obtained, the gradient of the pressure may be estimated by differentiating Eq. (17)

$$\nabla p(\vec{x}) \approx \sum_{J=1}^{N} \nabla\Phi_J(\vec{x})\,\hat{p}_J \tag{23}$$

The partial derivatives of the shape function with respect to x are found by differentiating Eq. (18) [see, e.g., Atluri and Zhu (1998)]. Thus,

$$\Phi_{J,x} = \psi^T_{,x}\mathbf{A}^{-1}\mathbf{B}_J + \psi^T\mathbf{A}^{-1}_{,x}\mathbf{B}_J + \psi^T\mathbf{A}^{-1}\mathbf{B}_{J,x} \tag{24}$$

where $\mathbf{A}^{-1}_{,x}$ is the partial derivative of $\mathbf{A}^{-1}$ with respect to x and is evaluated by $\mathbf{A}^{-1}_{,x} = -\mathbf{A}^{-1}\mathbf{A}_{,x}\mathbf{A}^{-1}$ with $\mathbf{A}_{,x} = \mathbf{B}_{,x}\Psi$; $\mathbf{B}_J$ is J-th column of Matrix $\mathbf{B}$ and its partial derivative is estimated by

$$\mathbf{B}_{J,x} = \frac{\partial w_J(\vec{x}-\vec{x}_J)}{\partial x}\psi(\vec{x}_J). \tag{25}$$

One can also obtain the partial derivative with respect to y (and z as well for 3D cases) in a similar way or by replacing x with y (or z) in Eqs. (24) and (25). It should be noted that although the base function is a linear function of coordinate variables $\vec{x}$, the shape function and its derivatives are nonlinear and continuous. The gradient of the pressure given by Eq. (23) is also continuous.

5 Imposing Boundary Conditions

Generally, there are two kinds of boundary conditions for the pressure in water wave problems, as shown in Eqs. (2a) and (2b): the free surface condition specifying the pressure (similar to essential boundary conditions in solid dynamics) and the solid boundary condition specifying the normal derivative of the pressure. The condition on the solid boundary was generally suggested to be imposed by applying Eq. (13) to incomplete circular or spherical sub-domains of boundary nodes [Atluri and Zhu (1998) and Atluri and Shen (2002)]. The implementation of an essential boundary condition is not very straightforward since the unknown nodal values in MLS approach to the shape function are not physical values as pointed out above and has received a considerable amount of research as summarized in Atluri (2004). The methods suggested therein include the penalty method, transformation method, collocation method and so on. These approaches works well for fixed (or with little movement) boundary problems as demonstrated by the cases in the publications, e.g. Atluri and Zhu (1998), Atluri and Shen (2002), Han and Atluri (2004a,b) and Sellountos and Polyzos (2003). As indicated by Ma (2005), the penalty method was tested for the nonlinear water wave problems and found that big errors, particularly near the free surface, can be quickly built up. The reason may be due to the fact that the addition of the penalty term inevitably produces some errors since it is not an exact representation of the free surface boundary condition. These errors may be negligible if the final solution could be found in one or a few time steps and if the boundary is not a part of solution but specified. However, they may accumulate to significant ones in several thousand time steps even though the whole scheme is stable in simulating nonlinear water waves. Although the accumulated errors may be reduced by shortening the length of the time step, it inevitably increases the computational costs. Therefore, Ma (2005) suggested, based on numerical tests, a better way to impose the free surface and solid boundary conditions in water wave problems is to use the following scheme:

$$\sum_{J=1}^{N} \Phi_J(\vec{x})\,\hat{p}_J = p_{atm} \text{ for nodes on the free surface} \tag{26a}$$

and

$$\sum_{J=1}^{N} \vec{n}\cdot\nabla\Phi_J(\vec{x})\,\hat{p}_J = \vec{n}\cdot\left(\vec{g}-\dot{\vec{U}}\right) \tag{26b}$$

for nodes on the solid boundary.

These work well for the formulation described by Ma (2005). For the MLPG_R method discussed here, numerical tests show that Eq. (26a) may be replaced by

$$p = p_{atm} \tag{26c}$$

with $\hat{p}_j$in Eqs. (17), (23) and (26b) equal p_{atm} if j-th node on the free surface.

6 Discretised Equations

Inserting Eq. (17) into Eq. (16) and combining the result with Equation (26) yields

$$\mathbf{K}\cdot\hat{\mathbf{P}} = \mathbf{F} \tag{27}$$

where

$$K_{IJ} = \begin{cases} \int\limits_{\partial\Omega_I} \Phi_J(\vec{x})\,\vec{n}\cdot\nabla\varphi ds - \Phi_J(\vec{x}_I) \\ \qquad\qquad \text{for inner nodes} \\ \vec{n}\cdot\nabla\Phi_J(\vec{x}_I) \\ \qquad\qquad \text{for nodes on solid boundaries} \end{cases} \tag{28a}$$

and

$$F_I = \begin{cases} \int\limits_{\Omega_I} \frac{\rho}{\Delta t}\vec{u}^{(*)}\cdot\nabla\varphi\, d\Omega \\ \qquad\qquad \text{for inner nodes} \\ \vec{n}\cdot\left(\vec{g}-\dot{\vec{U}}\right) \\ \qquad\qquad \text{for nodes on solid boundaries} \end{cases} \tag{28b}$$

where Node I denotes nodes that do not lie on the free surface; and Nodes J are those influencing Node I, determined by the weight function. Using Eq. (14), the boundary integral in Eq. (28a) can be simplified as:

$$\int\limits_{\partial\Omega_I} \Phi_J(\vec{x})\,\vec{n}\cdot\nabla\varphi ds = \frac{1}{2\pi\alpha R_I}\int\limits_{\partial\Omega_I} \Phi_J(\vec{x})\,ds \tag{29}$$

where α=1 for 2D and α=2 for 3D problems. The integral over the domain in Eq. (28b) can also be simplified and written as:

$$\int_{\Omega_I} \frac{\rho}{\Delta t}\vec{u}^{(*)}\cdot\nabla\varphi\, d\Omega = \frac{\rho}{2\pi\Delta t}\int_0^{2\pi}\int_0^{R_I} u_r^{(*)}(r,\theta)\, dr d\theta$$

for 2D cases; (30a)

$$\int_{\Omega_I} \frac{\rho}{\Delta t}\left(\vec{u}^{(*)}\cdot\nabla\varphi\right) d\Omega = \frac{\rho R_I}{4\pi\Delta t}\int_0^{R_I}\int_0^{2\pi}\int_0^{\pi} u_r^{(*)}\sin\theta\, dr d\theta d\beta$$

for 3D cases, (30b)

where $u_r^{(*)}$ is the radial component of $\vec{u}^{(*)}$.

7 Numerical Technique for domain integration

As shown in Eq. (30), the domain integration must be numerically evaluated, usually by using the Gaussian quadrature. To do so, more than 16 Gaussian points for 2D case and 64 Gaussian points for 3D cases may be required to obtain satisfactory results, at which the intermediate velocities are estimated by employing the MLS method. Evaluation of the velocities at so many points is time-consuming. In order to make the method more efficient, the following semi-analytical technique is suggested:

1. Dividing an integration domain into several subdomains;
2. Assuming intermediate velocities to linearly vary over each subdomain;
3. Performing the integration over each subdomain analytically.

The technique can be demonstrated by 2D cases. Let us consider a circular integration domain with a radius of R,centred at (x_0, y_0), as shown in Fig. 2 and divide it into four subdomains, 0-1-2, 0-2-3, 0-3-4 and 0-4-1. Over each subdomain, e.g., 0-1-2, the intermediate velocity components are assumed to be linear with respect to coordinates and given by

$$u^{(*)} = u_0^{(*)} + c_{ux}(x-x_0)/R + c_{uy}(y-y_0)/R \quad (31a)$$

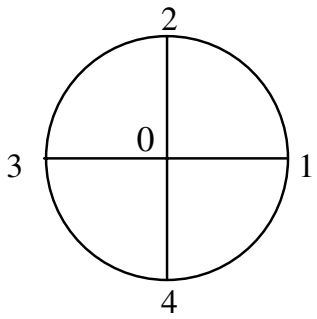


Figure 2 : Illustration of division of an integration domain.

$$v^{(*)} = v_0^{(*)} + c_{vx}(x-x_0)/R + c_{vy}(y-y_0)/R \quad (31b)$$

where $\left(u^{(*)}, v^{(*)}\right)$ are the intermediate velocity components at any point (x,y) in the subdomain 0-1-2; and $\left(u_0^{(*)}, v_0^{(*)}\right)$ are those at its centre. c_{ux}, c_{uy}, $c_{vx}$and c_{vy} are constants, which are determined in such a way that the velocity components equal to those at Point 1 and 2. Taking the $x-$component as an example, one should have

$$c_{ux}(x_1-x_0) + c_{uy}(y_1-y_0) = (u_1^{(*)} - u_0^{(*)})R$$
$$c_{ux}(x_2-x_0) + c_{uy}(y_2-y_0) = (u_2^{(*)} - u_0^{(*)})R$$

which gives

$$c_{ux} = \frac{\left(u_1^{(*)} - u_0^{(*)}\right)(y_2-y_0) - \left(u_2^{(*)} - u_0^{(*)}\right)(y_1-y_0)}{(x_1-x_0)(y_2-y_0) - (x_2-x_0)(y_1-y_0)}R \quad (32a)$$

$$c_{uy} = \frac{(x_1-x_0)\left(u_2^{(*)} - u_0^{(*)}\right) - (x_2-x_0)\left(u_1^{(*)} - u_0^{(*)}\right)}{(x_1-x_0)(y_2-y_0) - (x_2-x_0)(y_1-y_0)}R \quad (32b)$$

$c_{vx}$and c_{vy}can be found similarly or obtained by replacing $\left(u_1^{(*)}, u_2^{(*)}\right)$ with $\left(v_1^{(*)}, v_2^{(*)}\right)$ in Eq.(32); and thus the velocities in this subdomain are determined by Eq. (31). The velocities in other subdomains can also be estimated in this way. The only difference is that they may be related to the velocities at Points 3 and 4, depending on which subdomain is concerned. Consequently, the velocities at any points in the circle are determined by those at only five points (0, 1, 2, 3 and 4). Based on this, let us

consider the integration in Eq. (30a) over the circle. It can be rewritten as:

$$\int_0^{2\pi}\int_0^{R_I} u_r^{(*)}(r,\theta)\,drd\theta = \sum_{i=1}^{N_i}\int_{\vartheta_i}^{\vartheta_{i+1}}\int_0^{R_I} u_r^{(*)}(r,\theta)\,drd\theta$$

(N_i=4 for four divisions and$\vartheta_5 = \vartheta_1$).

The integration over each subdomain can be evaluated analytically using Eq. (31). For this purpose, Eq. (31) is rewritten as

$$u^{(*)} = u_0^{(*)} + c_{ux}\overline{r}\cos\theta + c_{uy}\overline{r}\sin\theta,$$

$$v^{(*)} = v_0^{(*)} + c_{vx}\overline{r}\cos\theta + c_{vx}\overline{r}\sin\theta,$$

and so

$$u_r^{(*)} = u_0^{(*)}\cos\theta + v_0^{(*)}\sin\theta + c_{ux}\overline{r}\cos^2\theta + c_{uy}\overline{r}\cos\theta\sin\theta + c_{vx}\overline{r}\cos\theta\sin\theta + c_{vy}\overline{r}\sin^2\theta. \quad (33)$$

where the transformation of$(x-x_0)/R = \overline{r}\cos\theta$ and $(y-y_0)/R = \overline{r}\sin\theta$ have been employed. It is easy to show that the integration of $u_0^{(*)}\cos\theta + v_0^{(*)}\sin\theta$ over the whole circle becomes zero and so can be omitted. The integration of other terms in Eq. (33) over Subdomain 0-1-2 is given by

$$\int_{\vartheta_1}^{\vartheta_2}\int_0^{R_I} u_r^{(*)}(r,\theta)\,drd\theta = \frac{1}{4}R_I\begin{bmatrix}(c_{ux}+c_{vy})(\vartheta_2-\vartheta_1)\\ +(c_{ux}-c_{vy})(\sin\vartheta_2\cos\vartheta_2-\sin\vartheta_1\cos\vartheta_1)\\ +(c_{uy}+c_{vx})\left(\sin^2\vartheta_2-\sin^2\vartheta_1\right)\end{bmatrix}.$$

Similar results can be obtained for other subdomains and the sum of these gives the results for the whole circular domain. Although the results depend on the number of subdomains, numerical tests show that four subdomains are good enough for 2D cases. It is envisaged that eight subdomains may be required for 3D cases.

It should be noted that with this semi-analytical technique, the velocities at only five points for a 2D circle with four divisions, instead of at more than 16 points if the Gaussian quadrature would be used, need to be evaluated. In 3D cases, the spherical domain may be divided in to 8 subdomains as indicated above and the velocities at only 7 points need to be evaluated, instead of at least 64 points when using the Gaussian quadrature. As a result, the reduction in CPU time spent on the evaluation of the domain integration is considerable.

The similar domain division technique was used in Atluri, Kim and Cho (1999) and Sellountos and Polyzos (2003). However, in their techniques, the Gaussian quadrature is still used for the integration over each subdomain but we do not use the Gaussian quadrature at all as demonstrated above.

8 Treatment of Pressure Gradient and/or Velocity

It is well known that backward or forward finite difference schemes approximating a derivative have lower order accuracy than a central scheme. Similarly, it can be understood that the pressure gradient estimated for nodes on or near boundaries using Eq. (23) and so the velocity using Eq. (11) may not be as accurate as for inner nodes because the related nodes to the boundary nodes distribute on one side only. In order to enhance the accuracy of results near the boundaries, particularly near the free surface, which is important for simulating water waves, Ma (2005) suggested that the pressure gradients after estimated by Eq. (23) be treated using the following equation

$$\nabla p(\vec{x}) \approx \sum_{J=1}^{N}\tilde{\Phi}_J(\vec{x})(\nabla p)_J \quad (34)$$

where the function $\tilde{\Phi}_J(\vec{x})$ is formulated by a similar method as for the shape function $\Phi_J(\vec{x})$ discussed above but using a different weight function. The weight function used for this purpose is the Gaussian weight function defined by

$$\tilde{w}_J(\vec{x}-\vec{x}_J) = \begin{cases}\frac{e^{-(\alpha_J\overline{r})^2}-e^{-\alpha_J^2}}{1-e^{-\alpha_J^2}} & \overline{r}=\frac{d_J}{r_J}\le 1\\ 0 & \overline{r}=\frac{d_J}{r_J}>1\end{cases} \quad (35)$$

where r_J and $d_J$have same meanings as before and α_J is an arbitrary coefficient controlling the shape of the weight function. The value of α_J may be chosen in a range of $(2\sim3)r_J$. Although the same weight function as in Eq. (22) may be adopted for this purpose, numerical tests show that it does not give as good results as Eq. (35) for the treatment of the pressure gradient.

In fact, the similar treatment can be applied to the velocity, rather than the pressure gradient. In this work, the

following formula is tested for computing velocity:

$$\vec{u}_m^{(n+1)}(\vec{x}) = (1-\gamma)\,\vec{u}^{(n+1)}(\vec{x}) + \gamma \sum_{J=1}^{N} \tilde{\Phi}_J(\vec{x})\,\vec{u}^{(n+1)}(\vec{x}_J) \tag{36}$$

where $\vec{u}_m^{(n+1)}(\vec{x})$ is the velocity at $\vec{x}$ used for updating while $\vec{u}^{(n+1)}(\vec{x})$ and $\vec{u}^{(n+1)}(\vec{x}_J)$ are, respectively, the velocities at $\vec{x}$ and $\vec{x}_J$ calculated by Eq.(11). γ is an artificial coefficient. When $\gamma = 0$, $\vec{u}_m^{(n+1)}(\vec{x}) = \vec{u}^{(n+1)}(\vec{x})$, i.e. no treatment is applied. When $\gamma = 1$, $\vec{u}_m^{(n+1)}(\vec{x}) = \sum_{J=1}^{N} \tilde{\Phi}_J(\vec{x})\,\vec{u}^{(n+1)}(\vec{x}_J)$, i.e. the velocity is fully smoothed. Based on preliminary numerical tests so far, it is suggested that the value of γ may be chosen in the range of $0.1 < \gamma < 0.3$. For the results presents in next section, it is taken as 0.2. It should be noted that with the modification in Eq. (36), the velocity,$\vec{u}_m^{(n+1)}(\vec{x})$, may not satisfy the continuity equation as long as$\gamma \neq 0$, implying that the fluid volume may not be conserved. Nevertheless, the results in Section 10 do not exhibit significant violence to the conservation. The reason behind this needs further investigations.

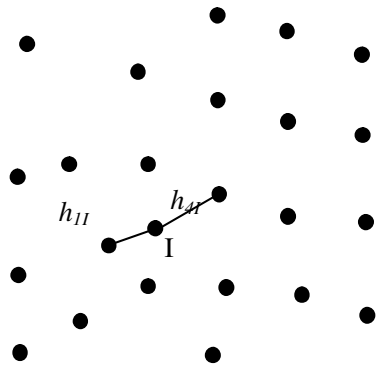


Figure 3 : Illustration of $h_{1I}$and h_{4I}

9 Sizes of integration and support domains

When implementing the MLPG_R method, one must determine the sizes of integration and support domains for all nodes. Theoretically, these sizes can be chosen arbitrarily. In practice, some factors have to be considered. The size of an integration domain is determined by R_I=εh_{1I}, where h_{1I} is the distance between Node I and its nearest node and ε is a coefficient. In order to ensure the integration domain is large enough but always located entirely inside the computational domain, the value of ε must be in the range of $0 < \varepsilon < 1$. The size of a support domain is given by $r_I = \kappa\, h_{4I}$, where h_{4I} is the distance between Node I and the fourth node when counting all neighbour nodes of Node I from the nearest to the farthest one, as shown in Fig. 3. In 2D cases with uniformly-distributed nodes,$h_{1I} = h_{4I}$. κ is the coefficient controlling the size of a support domain and its value should be specified with care. If κ is too small, the support domain is too small and too few nodes are involved in it, which may not be enough to result in a well-defined shape function. On the other hand if it is too large, the support domain is too big and too many nodes are contained in it. Too many nodes in one support domain require too much computational time to deal with. In addition, too large support domain may lower the accuracy. The reason for this is that the linear base function used to determine the shape function can give a good approximation to the pressure field only in a small local area. When the area is too large, the approximation is inevitably degraded. Based on numerical tests, it is found that the value for ε may be chosen in the range of 0.3 to 0.9; and the value for κ in the range of 1.5 to 3. The best values for a specific case need to be further explored. In this paper, all results presented in Section 10 are obtained using ε=0.6 and κ=1.5 to κ=2.0.

10 Numerical Validation

In this section, the numerical method described above is validated by applying it to water waves in a two dimensional tank generated by a wavemaker and by the motion of the tank. In the following discussion, all variables and parameters are non-dimensionalised using d and g, i.e.

$$(x,y,L,a) \to (x,y,L,a)\,d \quad t \to \tau\sqrt{\frac{d}{g}} \quad \omega \to \omega\sqrt{\frac{g}{d}}$$

10.1 Comparison with analytical solution for waves generated by a wavemaker with a motion of small amplitudes

The sketch of the wavemaker problem and coordinate system is illustrated in Fig. 4. The generated wave may be mono-chromatic or bi-chromatic depending on the motion of the wave maker $S(t)$. In order to reduce

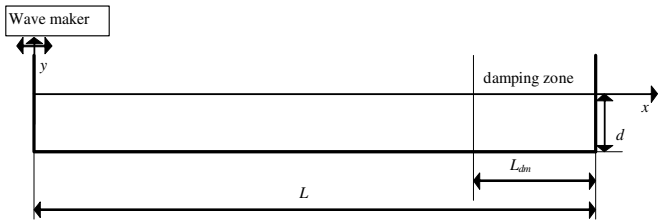


Figure 4 : Sketch of the problem and coordinate system

the reflection, a damping zone is added at the far end opposite to the wavemaker, in which the velocity ($\vec{u}_{dm}$) is modified by adding an artificial damping term and computed by using

$$\vec{u}_{dm}^{(n+1)} = \vec{u}_m^{(n+1)} - \nu(x)\ \vec{u}_m^{(n+1)}, \tag{37a}$$

where $\nu(x)$ is an artificial damping coefficient defined as

$$\nu(x) = \frac{1}{2}\nu_0 \left[1 - \cos\left(\frac{\pi(x-x_d)}{L_{dm}}\right)\right] \quad x \geq x_d \tag{37b}$$

where L_{dm} is the length of the damping zone taken as L_{dm}= $3d$; x_d is the$x$-coordinate of the left end of the damping zone; and ν_0 is the magnitude of the damping coefficient. A similar form of the damping coefficient was used in Ma, Wu and Eatock Taylor (2001a,b) for a numerical wave tank based on potential theory using a finite element method, but combined with a Sommerfeld condition. Extensive numerical tests were carried out in those papers to choose optimum values for ν_0. Those values are not necessarily optimum here as the formulation is different. Nevertheless, similar numerical investigations are not carried out in this paper since the main aim here is to validate the numerical method, instead of investigating the effectiveness of the damping zone. Instead, only one value is chosen based on some numerical tests, which is $\nu_0 = 0.1$. With this value, the reflection wave may not be eliminated completely, unless a long tank is used or simulations stop before reflection waves come into the area of interest.

Waves considered in this section are generated by a piston wavemaker in a tank with the length L=24, undergoing the following motion

$$S(t) = S_0(1 - \cos(\omega t)) \tag{38}$$

with the amplitude $S_0 = 0.01$ and the frequency ω =1.45. The resulting wave steepness defined by the ratio of wave heights from trough to crest to wave lengths is about 0.012. For such a small steepness, a linearised analytical solution may be found [Eatock Taylor, Wang and Wu (1994)] and can be used for validation. To do so, the characteristics of the relative error of numerical results are investigated. The relative error is defined as:

$$E_r = \frac{\|\eta_c - \eta_a\|}{\|\eta_a\|}$$

where $\|\eta\| = \int_{A_e} \eta^2 dA$ with A_e being area over which the error is estimated; η_c is the numerically computed elevation of wave profiles measured from the mean free surface and η_a is an analytical solution. The method used for estimating the relative error is very similar to that used in Atluri and Shen (2002) but the square root of the integral is not taken because the term $\int_{A_e} \eta^2 dA$, a measurement of wave energy, possesses clearer physical meaning than $\sqrt{\int_{A_e} \eta^2 dA}$ in water wave problems.

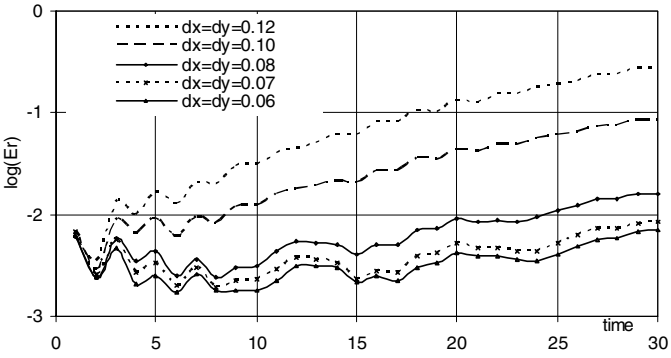


Figure 5 : Relative error at different time for different initial distance between nodes

The wave problem described above is simulated by using different number of nodes, uniformly distributed initially. Four cases are considered, in which the total number of nodes are N_t=201×9=1809 ($dx \approx dy \approx 0.12$, where dx and dy are the increment of distance between nodes in x and y direction, respectively), 241×11=2651, 151×13=3913 ($dx \approx dy \approx 0.08$), 343×15=5145 ($dx \approx dy \approx 0.07$) and 401×17=6817 ($dx \approx dy \approx 0.06$). The time step for these cases is chosen as Δt=0.02 based on similar investigations in Ma (2005). When estimating the relative error, A_e is chosen to be the area from the wavemaker to x=15; and the wave profiles from t=0 to t=30,

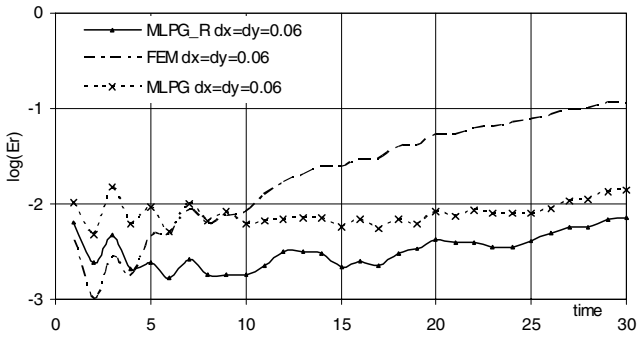


Figure 6 : Comparison of relative error of different methods

recorded at intervals of δt=1, are considered. Use of the smaller area rather than the whole area of the tank and shorter period (t$\leq$30) is to avoid the effects of the reflection on the error estimation. The relative errors evaluated in this way are presented in Fig. 5. This figure shows that the relative error varies with time and thus good numerical results at all time are more difficult to achieve for water waves than for other problems without free surfaces. Nevertheless, the error in numerical results obtained by the MLPG_R method is indeed reduced broadly with the decrease of distance between nodes, though the rate is different at different time. This fact implies that the numerical results with satisfactory accuracy at all time are achievable as long as a sufficiently large number of nodes are used.

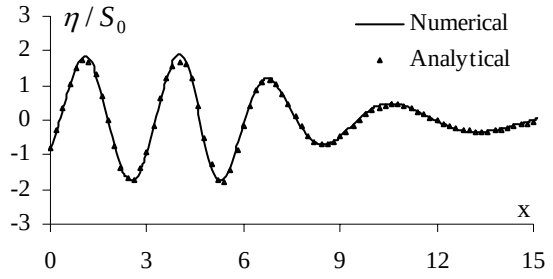
We also made investigations into the feature of convergence of three methods for waves associated with Fig.5: 1) MLPG method with the Heaviside step function being the test function [Ma (2005)]; FEM method based on a potential formulation [Ma, Wu and Eatock Taylor R. (2001a,b)]; and MLPG_R given in this paper. The rate of convergence of the three methods for the case with $dx \approx dy \approx 0.06$ or 6817 nodes is presented in Fig. 6. One can see from this figure that the relative error of MLPG_R is smaller than that of MLPG and also smaller than that of FEM except for the early period. One may then deduce that the number of nodes required in the MLPG_R method would be smaller than that in other two methods to gain the same accuracy of numerical results under the conditions specified here. However, it must be noted that the characteristics of convergence of numerical results for water wave problems depend on many factors, such as the wave frequency, the wave amplitude, the distribution of nodes and so on. Further investigations may be required before coming to conclude that the fact observed here generally holds in all situations.

To show the accuracy of numerical results from MLPG_R, wave profiles at three instants, which are obtained by using 6817 nodes (the fifth case in Fig. 5), are depicted and compared with the analytical solution in Fig.7. The agreement between the numerical and analytical results is very good, as can be seen. It implies that the method based on the MLPG_R described in this paper can produce accurate simulation of surface waves. It also implies that the volume of fluid is well conserved and the violence to continuity equation due to the modification in Eq. (36) does not pose a big problem, though reasons for this are not clear yet.

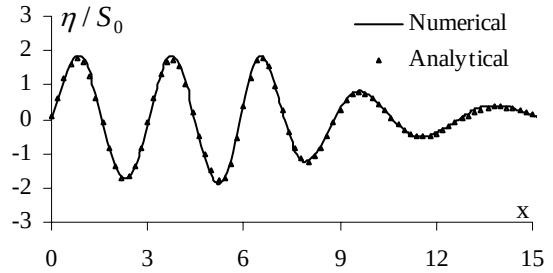
Comparison of the CPU time spent on each time step by the MLPG_R and MLPG method based on the Heaviside step function is preliminarily made in this work. It is found that the CPU time spent on the former is less than 50% of that on the latter.

10.2 Comparison with FEM results for waves generated by a wavemaker with a motion of larger amplitudes

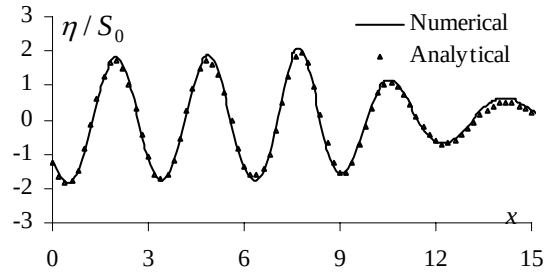
In the section, the results from the MLPG_R method are compared with those from the finite element method (FEM) in Ma, Wu and Eatock Taylor (2001a,b), which is based on the potential theory, different from the governing equations (Eqs. 1 and 2). The waves are still generated by the wavemaker moving as specified by Eq. (38) with the same frequency but with larger amplitudes. Two amplitudes are considered: one is S_0=0.05 and the other is S_0 =0.064; and so the waves have stronger nonlinearity, for which the linearised solution may not be valid. The total number of nodes used for these cases is 6817, which are again uniformly distributed initially. The number of nodes used for the finite element analysis is roughly similar. The comparison of results obtained by the two methods is given in Fig. 8 for S_0=0.05 and Fig. 9 for S_0 =0.064, respectively. In these figures, the dots represent the positions of nodes at the time shown on the upper-right corner obtained by the MLPG_R method. The free surface resulting from the finite element method is denoted by solid lines. As can be seen, the results from the two methods are very close, though the difference is visible. One of possible reasons for the difference may be due to the fact that the mathematical formulations in the



a) t=20



b) t=24



c) t=30

Figure 7 : Wave profiles at different times for S_0=0.01 and ω=1.45 and comparison with analytical solution

two methods are different as point above.

10.3 Comparison with analytical and experimental results for sloshing waves in a moving tank

Sloshing waves in a moving tank are associated with various engineering problems, such as liquid oscillations in large storage tanks caused by earthquakes, motions of liquid fuel in aircrafts and spacecrafts, liquid motions in containers and the water flow on decks of ships. The loads produced by wave motions of this kind can cause structural damage and the loss of the motion stability of objects such as ships. There has been a considerable amount of experimental, numerical and analytical work on wave sloshing. Nevertheless, the detail review on sloshing waves will not be given here, which may be found in Wu, Ma and Eatock Taylor (1998) and other papers cited therein. That is because the main purpose of this section is to further validate the MLPG_R method using some of experimental and analytical data in literature about sloshing waves.

Although one may here employ the same governing equations as in Eqs. (1) and (2) established for a fixed coordinate system, it is advantageous to use alternative ones established for a coordinate system moving together with the tank; and thus one can study the fluid flow relative to the tank, which is of primary interest. The sketch of the sloshing wave problem described in the moving coordinate system is still similar to that in Fig. 4 but without the wavemaker and damping zone. For such a system, the governing equations and boundary conditions for the problem are given as:

$$\nabla \cdot \vec{v} = 0 \tag{39a}$$

$$\frac{D\vec{v}}{Dt} = -\frac{1}{\rho}\nabla p + \vec{g} - \frac{D\vec{U}}{Dt} \tag{39b}$$

$$\frac{D\vec{r}}{Dt} = \vec{v} \text{ and } p = p_{atm} \text{ on the free surface} \tag{39c}$$

$$\vec{n} \cdot \vec{v} = 0 \text{ and } \frac{\partial p}{\partial n} = \rho\left(n \cdot \vec{g} - n \cdot \frac{D\vec{U}}{Dt}\right) \tag{39d}$$

on the walls and the bed of the tank.

where $\vec{v}$is the relative velocity of fluid and $\vec{U}$the velocity of the tank. These governing equations and boundary conditions are very similar to Eqs. (1) and (2). The only difference is that an extra term associated with the acceleration of the tank is contained in Eq. (39b), which reflects the effects of the motion of the tank on the relative velocity of fluid. Although these equations can be used for general motions of the tank, we only consider a motion in x-direction and defined by

$$\begin{aligned} X(t) &= X_0 \sin(\omega t) \\ U(t) &= \dot{X}(t) = X_0 \omega \cos(\omega t) \end{aligned} \tag{40}$$

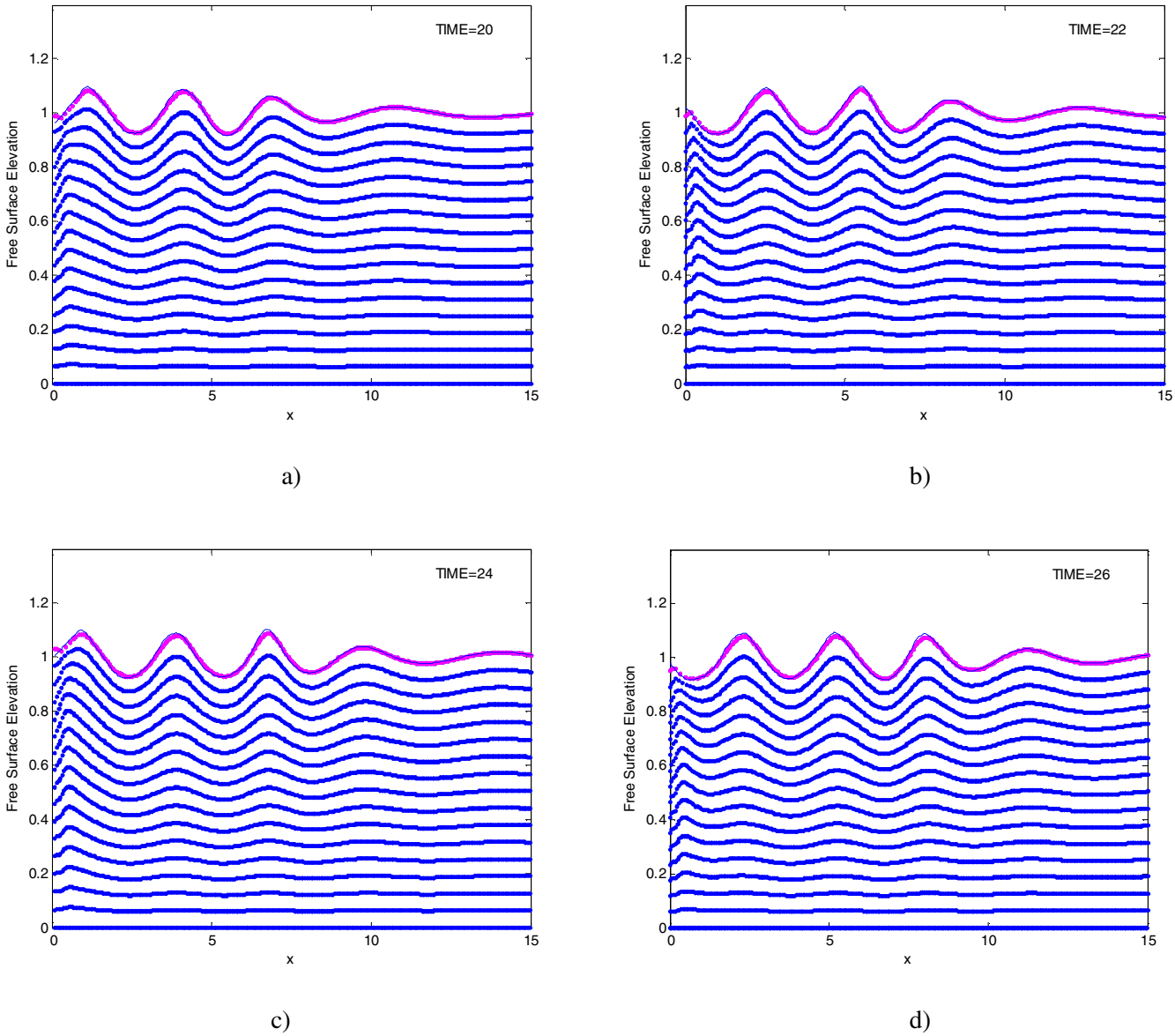


Figure 8 : Comparison of wave profiles obtained by MLPG_R and FEM for S_0=0.05 and ω=1.45. (Dots: node configuration obtained by MLPG_R; solid line: the free surface obtained by FEM)

where X_0 and ω are the amplitude and frequency ω, respectively. For such a motion, a linearised analytical solution [Faltinsen (1978)] can be found when wave amplitudes are small. Based on this solution, the dimensionless natural frequency of the tank is given by $\omega_{nj} = \sqrt{\frac{(2j+1)\pi}{L}\tanh\frac{(2j+1)\pi}{L}}$ for j=0,1,2.... When the frequency of motions equals one of these natural frequencies, the amplitude of the wave in the tank can grow infinitely in theory. Even when the frequency of the tank motion is not equal but near to the natural frequencies, the wave amplitudes can also grow to large ones. The numerical results from the MLPG_R method will be compared with this solution to further validate the method. For this purpose, the dimensionless length of the tank is taken as L=2, which gives

$$\omega_{n0} = \sqrt{\frac{\pi}{2}\tanh\frac{\pi}{2}}.$$

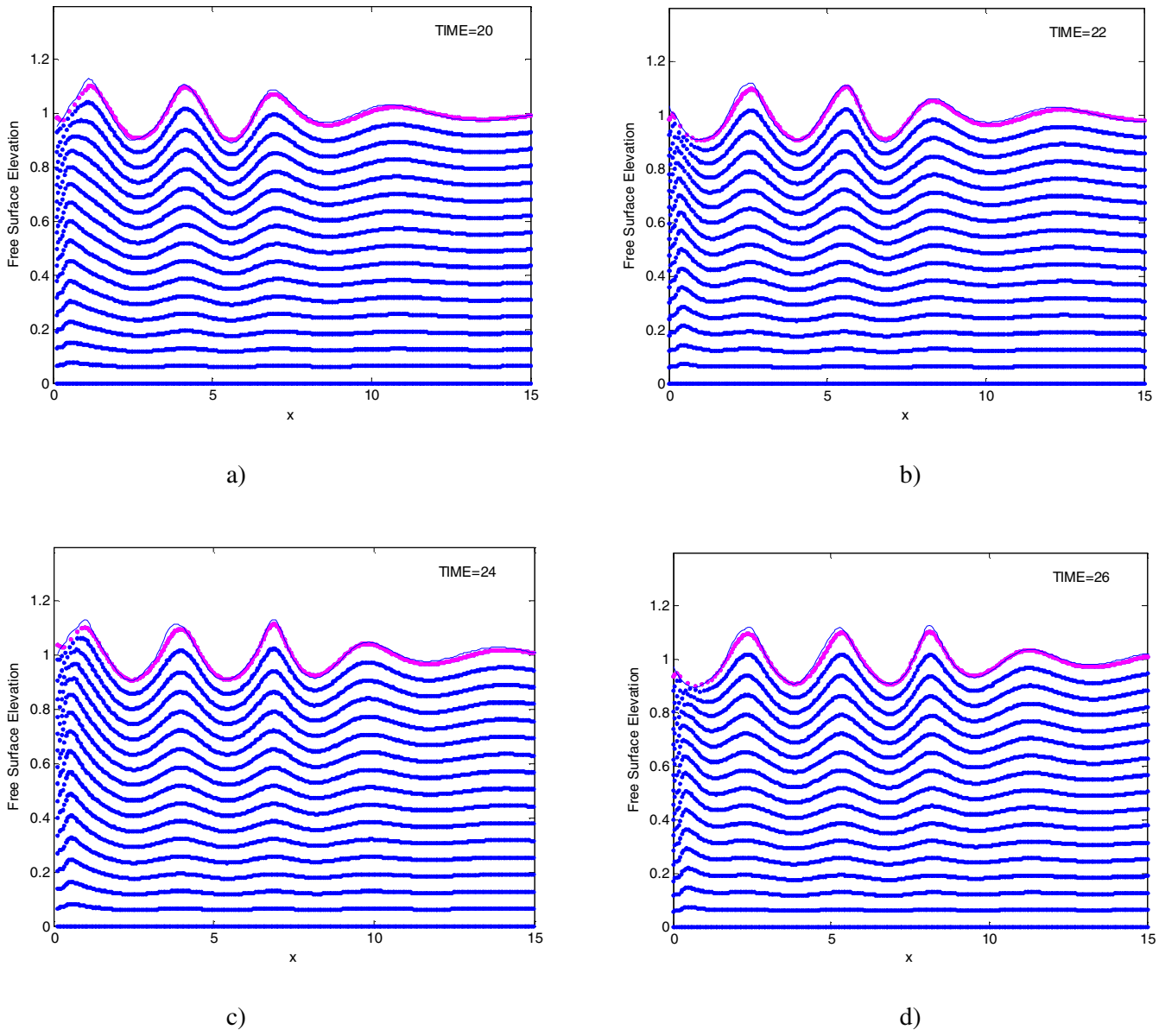


Figure 9 : Comparison of wave profiles obtained by MLPG_R and FEM for S_0=0.064 and ω=1.45.(Dots: node configuration obtained by MLPG_R; solid line: the free surface obtained by FEM)

Four cases are considered with the same amplitude of 0.002 but different frequencies of the tank motion, i.e. $\omega = 0.9\ \omega_{n0}$, $0.98\ \omega_{n0}$, $1.02\ \omega_{n0}$ and $1.1\ \omega_{n0}$. Convergent tests have been carried out and found that dx=dy=0.08 and 100 time steps (Δt = T/100, $T=2\pi/\omega$) in each period of the tank motion yield satisfactory solutions. The numerical time histories at x=0 (upper left corner of the tank) are depicted in Fig. 10 together with the analytical solution of Faltinsen (1978). The agreement between them is quite good as can be seen from this figure, implying that the MLPG_R can also produce accurate results for sloshing waves.

The present numerical method is now used to analyze a case with an amplitude of X_0=0.0186, which is about nine times larger than that in the previous case. The frequency of the motion is taken as $\omega/\omega_{n0} \approx 0.999$, very near to the first resonant frequency. The case with these parameters has been experimentally investigated by Okamoto and

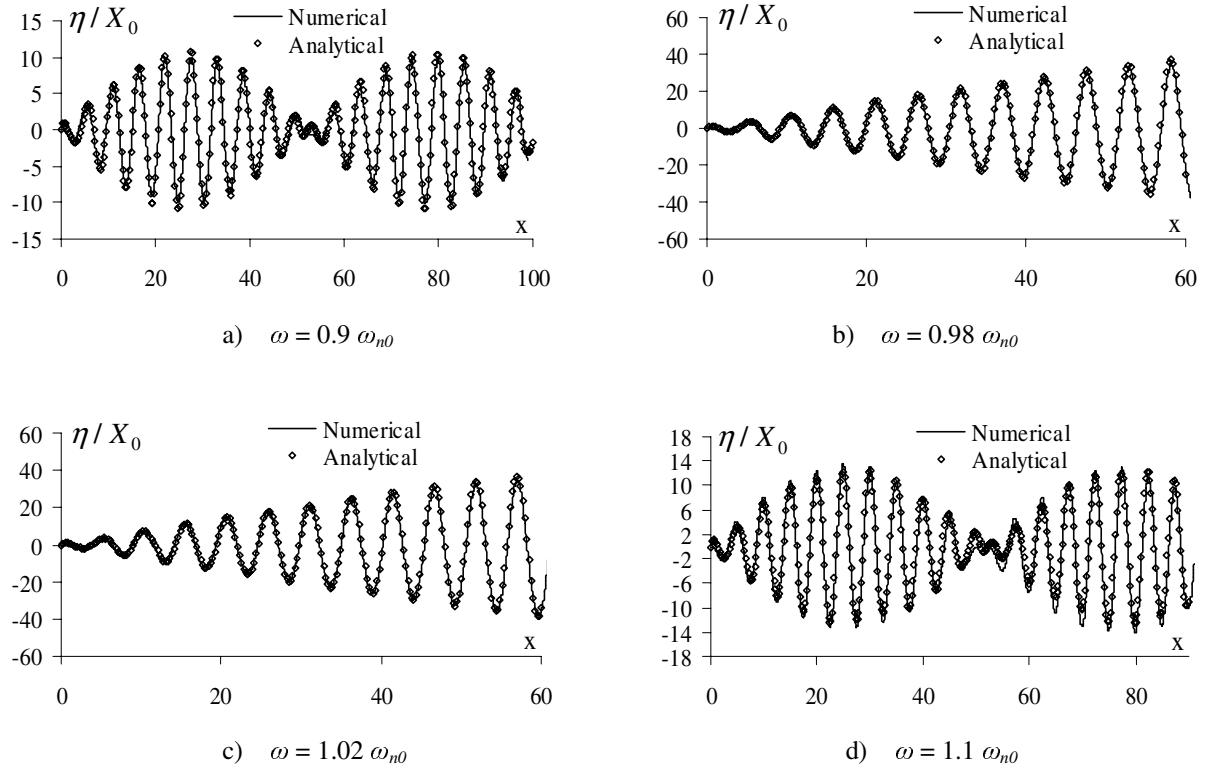


Figure 10 : Time histories of sloshing waves at x=0 for different frequencies (the analytical solution from Faltinsen (1978))

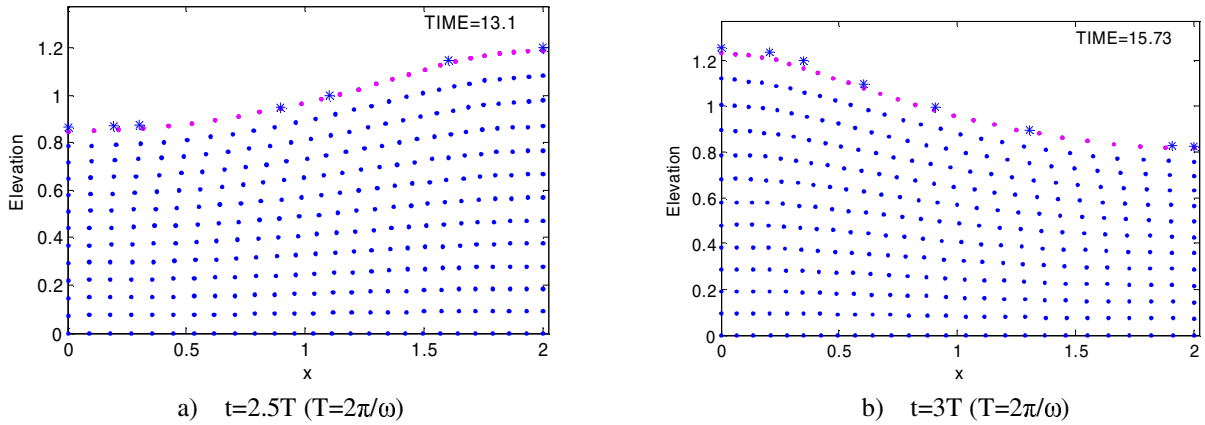


Figure 11 : Configurations of nodes at two instants for X_0=0.0186 and $\omega/\omega_{n0} \approx 0.999$ (*: experimental data from Okamoto and Kawahara (1990))

Kawahara (1990). To simulate this case, nodes are uniformly distributed initially with dx=dy=0.06 and the time step is also taken as Δt = T/100. Fig. 11 plots nodal configurations at two time instants, t=2.5T and 3T, respectively. The experimental data of Okamoto and Kawahara (1990) are also shown in this figure. Again, the free sur-

faces obtained by numerical simulation agree well with their experimental data.

11 Conclusions

In this paper, the MLPG_R method has been developed for simulating nonlinear water waves. This method is different from one described in Ma (2005) in that the solution for Rankine sources rather than the Heaviside step function is taken as the test function. Based on this test function, a weak form of governing equations has been derived, which does not contain the gradients of unknown functions and therefore makes numerical discretisation of the governing equations relatively easier and more efficient.

A semi-analytical technique is developed to evaluate the domain integral involved in this method. It can dramatically reduce the CPU time spent on the numerical calculation of this part.

The present method has been applied to two-dimensional waves generated by a wavemaker and by the surging motion of a tank. The numerical results from this method have been compared with analytical solutions, finite element results and experimental data in some cases. These comparisons demonstrate that the MLPG_R method can produce accurate results. The conclusion encourages us to apply the method to more complicated and practical problems in future work.

Acknowledgement: This work is sponsored by EPSRC, UK (GR/R78701), to which the author is most grateful.

References

Atluri, S.N. (2004): The Meshless Local Petrov-Galerkin (MLPG) Method for Domain & Boundary Discretizations, Tech Science Press.

Atluri, S.N.; Kim, H.G.; Cho, J.Y. (1999): A Critical Assessment of the Truly Meshless Local Petrov Galerkin (MLPG) and Local Boundary Integral Equation (LBIE) Methods, *Computational Mechanics*, 24:(5), pp. 348-372.

Atluri ,S.N.; Shen ,S. (2002): The Meshless Local Petrov-Galerkin (MLPG) Method: A Simple & Less-costly Alternative to the Finite Element and Boundary Element Methods, *CMES: Computer Modelling in Engineering & Sciences*, Vol. 3 (1), pp. 11-52.

Atluri, S.N.; Zhu, T. (1998): A New Meshless Local Petrov-Galerkin (MLPG) Approach in Computational Mechanics, *Computational Mechanics*, Vol. 22, pp. 117-127.

Atluri, S.N.; Zhu, T. (2000): New Concepts in Meshless Methods, *International J. Numerical Methods in Engineering*, Vol. 47 (1-3), pp. 537-556.

Batra, R. C.; Ching, H.-K. (2002): Analysis of Elastodynamic Deformations near a Crack/Notch Tip by the Meshless Local Petrov-Galerkin (MLPG) Method, *CMES: Computer Modeling in Engineering & Sciences*, Vol. 3 (6), pp. 717-730.

Belytschko, T.; Lu, Y. Y.; Gu, L. (1994): Element-Free Galerkin methods, *Int. J. Numr. Meth. Eng.*, Vol. 37, pp. 229-256.

Chorin, A.J. (1968): Numerical Solution of Navier-Stokes Equations, *Math. Computation*, Vol. 22, pp. 745-762.

Dalrymplem, R.A.; Knio, O. (2001): *SPH Modelling of Water Waves*, Proc. Coastal Dynamics, Lund.

Eatock Taylor; Wang, R; Wu, G.X. (1994): On the transient analysis of the wavemaker, 9^{th} *International Workshop on Water Waves and Floating Bodies*, Kuju, Oita, Japan.

Faltinsen, O.M. (1978): A numerical non-linear method of sloshing in tanks with two dimensional flow, *Journal of Ship Research*, Vol. 18(4), pp. 224–241.

Han, Z. D.; Atluri, S. N. (2004a): Meshless Local Petrov-Galerkin (MLPG) Approach for 3-Dimensional Elasto-dynamics, *CMC: Computers, Materials & Continua*, Vol. 1 (2), pp. 129-140.

Han, Z. D.; Atluri, S. N. (2004b): Meshless Local Petrov-Galerkin (MLPG) Approaches for Solving 3D Problems in Elasto-statics, *CMES: Computer Modeling in Engineering & Sciences*, Vol. 6 (2), pp. 169-188.

Idelsohn, S.R.; Storti, M.A.; Onate, E. (2001): Lagrangian formulations to solve free surface incompressible inviscid fluid flows, *Comput. Methods. Appl. Mech. Engrg.*, Vol. 191, pp. 583-593.

Koshizuka, S.; Nobe, A; Oka, Y. (1998): Numerical Analysis of Breaking Waves Using the Moving Particle Semi-Implicit Method, *Int J Numer Meth in Fluids*, Vol. 26, pp. 751–69.

Lin, H.; Atluri S.N. (2000): Meshless Local Petrov-

Galerkin (MLPG) method for convection-diffusion problems, *CMES: Computer Modelling in Engineering & Sciences*, Vol. 1 (2), pp. 45–60.

Lin, H.; Atluri S.N. (2001): The Meshless Local Petrov-Galerkin (MLPG) method for solving incompressible Navier-Stokes equations, *CMES: Computer Modeling in Engineering & Sciences*, Vol. 2 (2), pp. 117-142.

Liu, W. K.; Chen , Y.; Jun, S.; Chen, J. S.; Belytschko, T.; Pan, C.; Uras, R. A.; Chang, C. T. (1996): Overview and Applications of the Reproducing Kernel Particle Methods, *Archives of Computational Methods in Engineering: State of the art reviews*, Vol. 3, pp. 3-80.

Lo, E.Y.M.; Shao, S. (2002): Simulation of near-shore solitary wave mechanics by an incompressible SPH method, *Applied Ocean Research*, Vol. 24 (5), pp. 275-286.

Ma, Q.W. (2005): Meshless Local Petrov-Galerkin Method for Two-dimensional Nonlinear Water Wave Problems. *Journal of Computational Physics*, Vol. 205, Issue 2, pp. 611-625.

Ma, Q.W., and Yan, S. (2005) Quasi ALE Finite Element Method for Nonlinear Water Waves, accepted for publication by *J. of Computational Physics*.

Ma, Q.W; Wu, G.X.; Eatock Taylor R. (2001a): Finite element simulation of fully nonlinear interaction between vertical cylinders and steep waves–part 1 methodology and numerical procedure, *Int. J. Numer. Meth. Fluids*, Vol. 36, pp. 265-285.

Ma, Q.W; Wu, G.X.; Eatock Taylor R. (2001b): Finite element simulation of fully nonlinear interaction between vertical cylinders and steep waves–part 2 numerical results and validation, *Int. J. Numer. Meth. Fluids*, Vol. 36, pp. 287 – 308.

Monaghan, J.J. (1994): Simulating free surface flows with SPH, *Journal of Computational Physics*, Vol. 110, pp. 399-406.

Nayroles, B.; Touzot, G.; Villon P. (1992): Generalizing the Finite Element Method, Diffuse Approximation and Diffuse Elements, *Computational Mechanics*, Vol. 10, pp. 307-318.

Okamoto, T; Kawahara, M. (1990): Two-dimensional sloshing analysis by Lagragian finite element method. *International Journal for Numerical Methods in Fluids*, Vol. 11, pp. 453–477.

Onate, E.; Idelsohn, S. R.; Pin, F. Del; Aubry, R. (2004): *The Particle Finite Element Method — An Overview,* International Journal of Computational Methods, Vol. 1 (2), pp. 267-308.

Sellountos, E. J.; Polyzos, D. (2003): A MLPG (LBIE) method for solving frequency domain elastic problems, *CMES: Computer Modeling in Engineering & Sciences*, Vol. 4 (6), pp. 619-636.

Zienkiewicz, O.C.; Taylor, R.L. (2000), The Finite Element Method - Volume 3: Fluid Dynamics,5^{th} edition, Butterworth-Heinemann.

Zhu, T.; Zhang, J.D.; Atluri, S.N. (1998): A Local Boundary Integral Equation (LBIE) Method in Computational Mechanics, and a Meshless Discretization Approach, *Computational Mechanics*, Vol. 21, pp. 223-235.

Wu, G.X., Ma, Q.W and Eatock Taylor, R. (1998): Numerical simulation of sloshing waves based on finite element method, *Applied Ocean Research*, Vol. 20, pp. 337-355.