Copyright © 2008 ICCES Proceedings of ICCES'08, pp.1-6

Effects of Different Weight Functions on Convergent Property of SFDI

Q.W. Ma[1]

Summary

In the MLPG_R (Meshless Local Petrove-Galerkin based on Rankine source solution) method, one needs a meshless interpolation scheme for an unknown function to discretise the governing equation. A new scheme, called simplified finite difference interpolation (SFDI) has been recently devised by the author, which is more efficient and has better convergent property than others. This paper will discuss about how the convergent property is affected by different weight functions.

Introduction

In [1], the MLPG (Meshless Local Petrove-Galerkin) method (see, e.g., [2][3][4]) was developed into a new form called the MLPG_R method, suitable for modelling nonlinear water waves. In the MLPG_R method, the water wave problems are solved using a time-step marching procedure, in which the velocities of particles at each time step are updated by numerical integration and the pressure is found by solving a boundary value problem. To do so, one needs a meshless interpolation scheme. Computational costs spent on them are considerable. Reducing these costs can make the method more efficient. The MLS (moving least squqre) method was utilised by the author in [1]. However, the MLS method has some drawbacks. Ma [5] recently devised another new scheme called SFDI - Simplified Finite Difference Interpolation. Based on numerical tests, it was shown that this scheme had the following advantages. (1) It is as accurate as the MLS method generally but more accurate than the later when the number of neighbour nodes was small. (2) It needs considerably less CPU time than the MLS method. (3) The scheme is not sensitive to scale factors. The numerical tests in [5] are based only on the spline weight function. One may ask if the above statements hold when using other weight functions possessing different properties. This paper is to investigate the effects of different weight functions on the property of the SFDI to answer the question.

Brief of Simplified Finite Difference Interpolation (SFDI) Scheme

The SFDI scheme includes the expressions for the meshless interpolation of an function and its gradient. The detail derivations have been described in [5]. Only the main expressions are summarised here.

Function interpolation

A general function $f(\vec{r})$ at a point $\vec{r}_0$ is expressed, in terms of its nodal values

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

at a set of nodes, by:

$$f(\vec{r}_0) = \sum_{J=1}^{N} \Phi_J(\vec{r}_0;\vec{r}_I) f(\vec{r}_J), \tag{1a}$$

where N is the number of nodes affecting the point, $\vec{r}_J$ is the position vector of Node J and $\Phi_J(\vec{r}_0;\vec{r}_I)$ is the shape function in the SFDI interpolation and is defined by

$$\Phi_J(\vec{r}_0;\vec{r}_I) = \frac{w(|\vec{r}_J - \vec{r}_0|)}{\sum_J^N w(|\vec{r}_J - \vec{r}_0|)} - (1-\delta_{IJ}) B_{0,J}(\vec{r}_I) + \delta_{IJ} \sum_{J\neq I}^{N} B_{0,J}(\vec{r}_I), \tag{1b}$$

where

$$\delta_{IJ} = \begin{cases} 1 & I = J \\ 0 & I \neq J \end{cases}, \quad B_{0,J}(\vec{r}_I) = \frac{w(|\vec{r}_J - \vec{r}_I|)}{|\vec{r}_J - \vec{r}_I|^2} \sum_{k=1}^{d} \frac{\vec{R}_{0,x_k}}{n_{I,x_k}} (\vec{r}_{J,x_k} - \vec{r}_{I,x_k})$$

and

$$\vec{R}_{0,x_k} = \sum_J^N (\vec{r}_{J,x_k} - \vec{r}_{0,x_k})\, w(|\vec{r}_J - \vec{r}_0|) \Big/ \sum_J^N w(|\vec{r}_J - \vec{r}_0|).$$

In the above equations, $w(|\vec{r}_J - \vec{r}_0|)$ is a weight function which will be given below and d is the number of spatial dimensions (d=2 for 2D cases and d=3 for 3D cases) and x_k for k=1,2 and 3 is x, y and$z$, respectively. It can be seen that Eq. (1) does not need to solve any algebraic equation or to calculate the inverse of a matrix. That is why it is more efficient than the MLS method.

Gradient interpolation

The expression for the gradient is given by

$$(f_{,x_m})_{\vec{r}_0} = \sum_{J=1}^{N} \tilde{\Gamma}_{mJ}(\vec{r}_0)\,[f(\vec{r}_J) - f(\vec{r}_0)] \tag{2a}$$

with

$$\tilde{\Gamma}_{mJ}(\vec{r}_0) = \sum_{k=1}^{d} \tilde{A}_{mk} C_{kJ}, \quad J = 1,2,3\cdots. \tag{2b}$$

where

$$C_{mJ} = \frac{1}{n_{0,x_m}} \frac{(\vec{r}_{J,x_m} - \vec{r}_{0,x_m})}{|\vec{r}_J - \vec{r}_0|^2} w(|\vec{r}_J - \vec{r}_0|)$$

and $\tilde{A}_{mk}$ is an element in the matrix $\left[\tilde{A}\right]$, the inverse matrix of $[A]$, i.e. $\left[\tilde{A}\right] = [A]^{-1}$, with the later defined by

$$A_{mk} = a_{0,mk} \quad (m \neq k) \quad \text{and} \quad A_{mm} = 1.$$

where

$$a_{0,mk} = \frac{1}{n_{0,x_m}} \sum_{J}^{N} \frac{\left(\vec{r}_{J,x_m} - \vec{r}_{0,x_m}\right)\left(\vec{r}_{J,x_k} - \vec{r}_{0,x_k}\right)}{\left|\vec{r}_J - \vec{r}_0\right|^2} w\left(\left|\vec{r}_J - \vec{r}_0\right|\right)$$

.

Although Eq. (2) needs to evaluate inverse of the matrix$[A]$, the size of this matrix is smaller than that of its counterpart in the MLS method. The size in this method is 2×2 for 2D cases or 3×3 for 3D cases while it is 3×3 for 2D cases or 4×4 for 3D cases in the later. In addition, the number of matrixes in the multiplication is also smaller. Consequently, Eq. (2) require less CPU time than the MLS method.

Weight Functions

Three different weigh functions will be considered in this paper. They are defined as follows. In the definitions, $|\Delta\vec{r}|$ is the distance of two points and h_I is the size of the support domain of the weight function which is given by $h_I = \kappa h_{4I}$ with h_{4I} being the distance between Node I and the fourth nearest neighbour node and with κ being a scale factor.

(1) Gaussian weight function (WF1)

$$w\left(|\Delta\vec{r}|\right) = \begin{cases} \dfrac{\exp\left[-\left(\left|\Delta\vec{r}\right| \Big/ c_I\right)^2\right] - \exp\left[-\left(h_I \Big/ c_I\right)^2\right]}{1-\exp\left[-\left(h_I \Big/ c_I\right)^2\right]} & 0 \le \frac{|\Delta\vec{r}|}{h_I} \le 1 \\ 0 & \frac{|\Delta\vec{r}|}{h_I} > 1 \end{cases}$$

where c_I is a node-dependent constant controlling the shape of the weight function and is taken as the distance between Node I and the fifth nearest neighbour in this paper. It is well known that this function is C^0-continuous.

(2) Spline weight function (WF2)

$$w\left(|\Delta\vec{r}|\right) = \begin{cases} 1 - 6\left(\frac{|\Delta\vec{r}|}{h_I}\right)^2 + 8\left(\frac{|\Delta\vec{r}|}{h_I}\right)^3 - 3\left(\frac{|\Delta\vec{r}|}{h_I}\right)^4 & 0 \le \frac{|\Delta\vec{r}|}{h_I} \le 1 \\ 0 & \frac{|\Delta\vec{r}|}{h_I} > 1 \end{cases}$$

This function is C^1-continuous.

(3) Simple weight function (WF3)

$$w\left(|\Delta\vec{r}|\right) = \begin{cases} \frac{h_I}{|\Delta\vec{r}|} - 1 & 0 \le \frac{|\Delta\vec{r}|}{h_I} \le 1 \\ 0 & \frac{|\Delta\vec{r}|}{h_I} > 1 \end{cases}$$

This one has a singular point ($|\Delta\vec{r}| = 0$). When using it, the singular point should be excluded.

As indicated above, the three functions have very different properties from each other and may be considered as good representatives of various weight functions often used in the community who employs meshless methods.

Numerical Results

In this section, investigations will be made into the effects of three weight functions on the convergent rate of the new shape functions in Eqs. (1) and (2). The computational domain for the tests is chosen as a square with the length of sides being 1. The nodes, at which the nodal values $f(\vec{r}_J)$ are defined, are irregularly distributed by using quasi-random number; see, for example, [6]. A typical node distribution is illustrated in Fig 1.

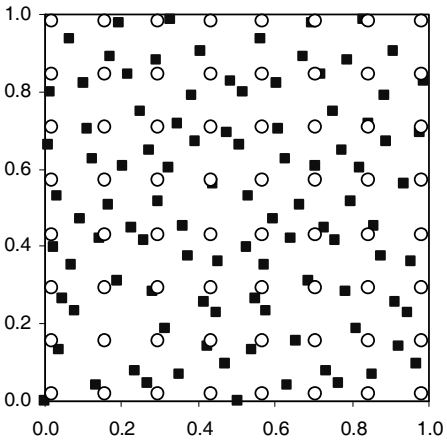

Figure 1: Nodes and interpolation points (O: interpolation points; ■: nodes)

Generally, Eq. (1) is used in the MLPG_R method to discretise the governing equation for pressure, giving the relationship between pressure at any point and its nodal values. On the other hand, Eq. (2) is employed to estimate the pressure gradient at nodes in terms of its nodal values for evolving the velocity. Therefore, different approaches should be adopted when investigating the convergent rate of Eq. (1) and Eq. (2). To investigate the convergent rate of Eq. (1), the assumed functions are interpolated by the equation at a set of points. These interpolation points are not necessarily coincided with any node and the number of the points is not necessary the same as the number of nodes. In fact, the number of points in the following tests is fixed as 64. For clarity, these points are denoted by $\vec{r}_{0k}$ while nodes are still denoted by $\vec{r}_J$ as above. The distribution of the points is also illustrated in Fig. 1 together with nodes. The error of the results of interpolation is defined as

$$E_{mean} = \frac{1}{K}\sum_{k}^{K}\left|f(\vec{r}_{0k}) - \bar{f}(\vec{r}_{0k})\right| \tag{3}$$

where E_{mean} is the mean error, $f(\vec{r}_{0k})$ is the interpolated value at the interpolation points and $\bar{f}(\vec{r}_{0k})$ is the values computed by the formula defining a function at the same point; K is the number of interpolation points. For investigating the convergent rate of Eq. (2), the gradient of the function is estimated at the nodes by the

equation and the error is found by the following expression

$$E_{gmean,y} = \frac{1}{M}\sum_{J}^{M}\left|f_{,y}(\vec{r}_J) - \bar{f}_{,y}(\vec{r}_J)\right| \tag{4}$$

where $E_{gmean,y}$ represents the mean error of the y-component of the gradient, $f_{,y}(\vec{r}_J)$ is the approximate value of the y-component of the gradient obtained by interpolation, $\bar{f}_{,y}(\vec{r}_J)$ is its counterpart computed by the formula defining the function and M is the total number of nodes. The mean error of the x-component of the gradient can be estimated in the same way by replacing y with x.

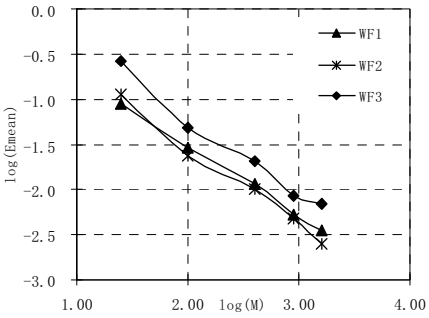

Fig. 2b (χ=1.8)

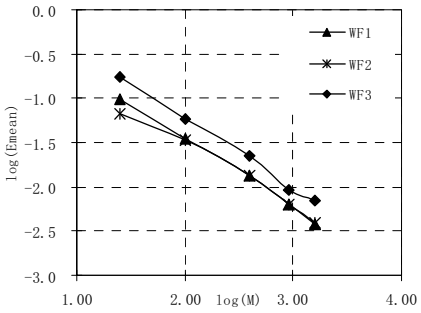

Fig. 2c (χ=2.5)

Figure 2: Convergent rates for interpolating the polynomial function by using three weight functions.

For numerical tests, a polynomial function of second order expressed by $f = 1 + 2x^2 + 3y^2$ is considered. Its gradient components are easy to find and is not written out here. The total number (M) of nodes in the square domain described above is selected as 25, 100, 400, 900 or 1600, respectively, in different runs. The convergent rates for interpolation of the function at 64 points by using three different weight functions are shown in Fig. 2, where the scale factors are 1.8 and 2.5, respectively. The convergent rates are denoted by the decrease of mean errors defined in Eq. (3) with the increase in the total number of nodes. The figure demonstrates that the Gaussian (WF1) and the Spline (WF2) weight functions lead to almost the same results and are better than those given by using the simple weight function (WF3).

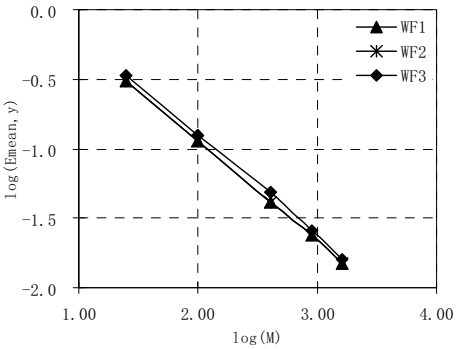

Figure 3: The convergent rate for estimating the gradients of the polynomial function using three weight functions (χ=2.5)

Fig. 3 shows the convenient rates for estimating the gradient components or partial derivative of the polynomial function. The value of $f_{,y}$ is obtained by using three weight functions for the scale factor κ =2.5. It can be seen that the different weight functions give almost the same convergent rate. The similar cases are investigated for other scale factors and also for a cosine function but not shown here to save the space. The property of the convergence is very similar to what we see in Figs. 2 and 3.

Conclusion

In this paper, the effects of three different weight functions on the convergent property of a new meshless interpolation scheme (named as the SFDI) are investigated. These weight functions have quite different continuous behaviours. The numerical results show that the convergent property of the SFDI is not significantly different for the different weight functions. This may be considered as another good property as one may not know which weight function is the best unless carrying out extensive numerical tests.

Acknowledgement

This work is sponsored by The Leverhulme Trust, UK, to which the author is most grateful.

References

1. Ma, Q.W. (2005): MLPG Method Based on Rankine Source Solution for Simulating Nonlinear Water Waves, submitted to *CMES* for publication.
2. Atluri, S.N.; Zhu, T. (1998): A New Meshless Local Petrov-Galerkin (MLPG) Approach in Computational Mechanics, *Computational Mechanics*, Vol. 22, pp. 117-127.
3. Atluri, S.N.; Liu, H.T.; Han, Z.D. (2006): Meshless Local Petrov-Galerkin (MLPG) Mixed Finite Difference Method for Solid Mechanics, *CMES: Computer Modeling in Engineering & Sciences*, Vol. 15, No.1, pp. 1-16.
4. Atluri, S.N. (2005): Methods of Computer Modeling in Engineering and the Sciences. Vol. 1, Tech Science Press.
5. Ma, Q.W. (2007): A New Meshless Interpolation Scheme for MLPG_R Method, accepted by *CMES: Computer Modeling in Engineering & Sciences*.
6. Faure, H. (1990): Using permutations to reduce discrepancy. *J. Comp. Appl. Math.*, 31:97-103.