European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 4, no. 3-4, 2009, pp. 193-201
ISSN 1790–8140

An Efficient Scheme for Meshless Analysis Based on Radial Basis Functions [1]

S. Nakata[2]

Institute of Science and Engineering,
Ritsumeikan University,
Kusatsu, Shiga, 525-8577 Japan

Received 29 January, 2007; accepted in revised form __ __, 2009

Abstract: A meshless method based on radial point interpolation was recently developed as an effective tool for solving partial differential equations, and has been widely applied to a number of different problems. In addition to the primary advantage of the meshless methods that the computation is performed without any connectivity information between field nodes, the radial point interpolation-based meshless method has several advantages such as the stability of the shape functions and simple implementation of boundary condition enforcement. This paper introduces a new scheme for the radial point interpolation-based meshless method. This method enables fast computation by modifying the construction and evaluation of the shape functions. Numerical examples are also presented to show that a reliable solution can be obtained with low computational cost.

Keywords: Partial Differential Equations, Meshless Method, Radial Basis Function, Radial Point Interpolation

Mathematics Subject Classification: 65N30

1 Introduction

In recent years, the meshless method has been widely used to solve partial differential equations (PDEs). The element-free Galerkin method (EFG) proposed by Belytschko et al. [1] is one such meshless PDE solver. This method enables construction of shape functions using the technique of moving least squares (MLS) approximation [2], which does not require node connectivity information for constructing interpolating functions of distributed data. See [3] for more information.

The radial point interpolation method (RPIM) proposed by Wang and Liu [4] is another important meshless approach for boundary value problems. This method has been applied to many kinds of actual problems such as 2- and 3-D solid analysis. See [4, 5, 6, 7] for more information. One remarkable advantage of this method is that the shape functions possess the Kronecker delta function property, which enables simple enforcement of Dirichlet boundary conditions. On the other hand, evaluation of the RPIM shape functions involves high computational cost, since the

[1]Published electronically December 10, 2009
[2]E-mail: snakata@is.ritsumei.ac.jp

shape functions are calculated as the solution of a linear system, and the coefficient matrix of the linear system varies according to the point of interest. As a result, this can be a computational bottleneck, particularly when the scale of the problem is large.

In this paper, we propose a method that enables fast evaluation of the RPIM shape functions to overcome the above-mentioned shortcoming. The main idea for fast computation is as follows:

- Divide a bounding rectangle of the problem domain into small rectangular subdomains and, for each subdomain, define a matrix for shape function construction that is constant in the subdomain. The matrix can be implemented in the same idea as the RPIM, with small modification of the definition of the shape functions.

- Compute decomposed forms of the matrices defined at all subdomains as pre-processing. The decomposed form, such as LU decomposition, contributes to a reduction of computational cost in solving linear systems.

Note that the matrices corresponding to all the subdomains are independent of the position where the shape functions are evaluated: however, the matrices vary depending on the position in RPIM. In this way, the matrices and the decomposed forms can be computed as a pre-processing—the shape functions can be evaluated efficiently by solving the linear system using the decomposed forms obtained in the pre-processing.

2 Construction of shape functions

Let Ω be a bounded domain in $\mathbb{R}^2$ and consider the boundary value problem,

$$\begin{cases} -\Delta u = f & \text{in} \quad \Omega, \\ u = \bar{u} & \text{on} \quad \Gamma_{\mathrm{D}}, \quad \dfrac{\partial u}{\partial n} = \bar{q} \quad \text{on} \quad \Gamma_{\mathrm{N}}, \end{cases}$$

where f is a field function given in Ω, $\bar{u}$ is a Dirichlet condition given on Γ_{D}, $\bar{q}$ is a Neumann condition given on Γ_{N} and $\partial/\partial n$ is differentiation along the outer normal to Γ_{N}.

We assume that N field nodes, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, distributed in the problem domain and on its boundary are given. Consider an approximation of the solution $u(\boldsymbol{x})$ represented as follows:

$$u(\boldsymbol{x}) \approx \sum_{i=1}^{N} u_i \phi_i(\boldsymbol{x}), \tag{1}$$

where $\phi_i(\boldsymbol{x})$ is a shape function corresponding to each field node $\boldsymbol{x}_i$. The purpose here is to determine the unknown coefficients, u_i, in (1), so that the approximate solution satisfies the boundary conditions. These coefficients can be obtained as the solution of the linear system,

$$A \begin{bmatrix} u_1 & \cdots & u_N \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix}^{\mathrm{T}}, \tag{2}$$

$$a_{ij} = \int_{\Omega} \nabla \phi_i(\boldsymbol{x}) \cdot \nabla \phi_j(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \quad y_i = \int_{\Omega} f(\boldsymbol{x}) \phi_i(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int_{\Gamma_{\mathrm{N}}} \bar{q} \phi_i(\boldsymbol{x}) \mathrm{d}l, \tag{3}$$

after enforcement of the Dirichlet boundary condition. In this section, we suggest a new definition of the shape function as a modification of the definition of shape functions in RPIM.

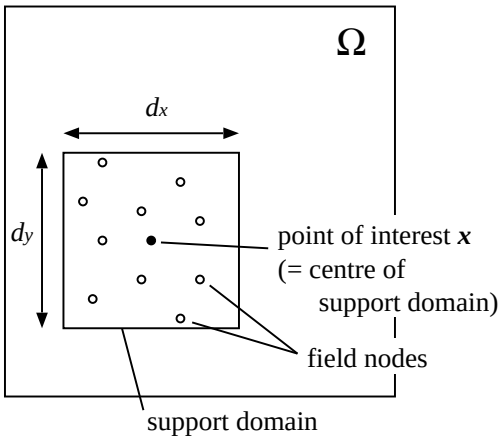


Figure 1: Support domain for RPIM shape function

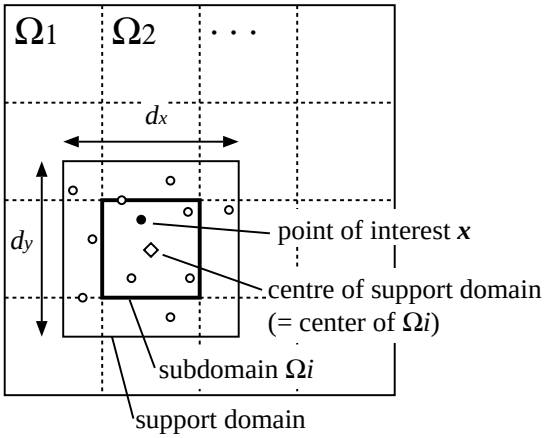


Figure 2: Support domain for modified RPIM shape function

2.1 Shape function used in RPIM

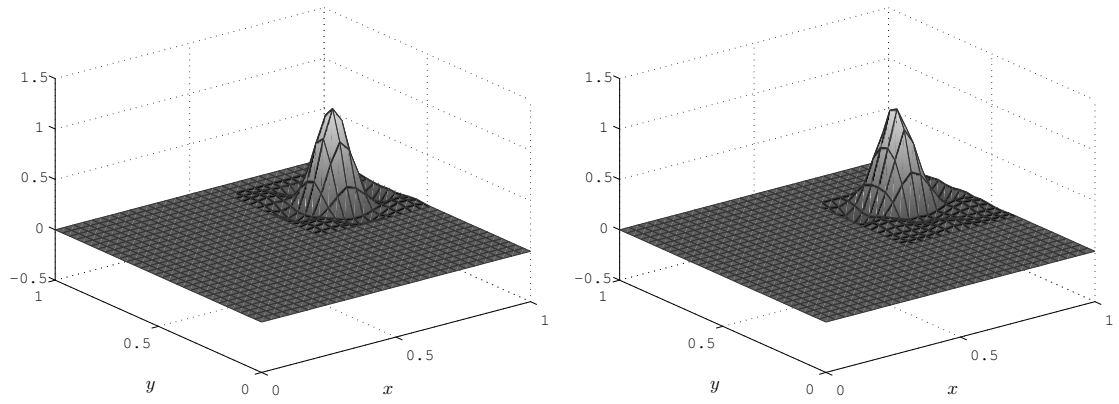
The shape functions $\phi_i(\boldsymbol{x})$ used in the original RPIM are defined using a support domain centred at a point of interest $\boldsymbol{x}$ [4]. An example of the rectangular support domain is shown in Fig. 1. The size of the support domain, d_x and d_y, are determined as

$$d_x = \alpha_x d_{cx}, \quad d_y = \alpha_y d_{cy}, \tag{4}$$

where d_{cx} and d_{cy} are average distances of the field nodes in x and y directions, respectively, and α_x and α_y are user-defined parameters determining the size of the support domain. Note that elliptic support domains used in [4] are also applicable.

Let the size of the support domain in x and y directions be d_x and d_y, respectively (as shown in the same figure), n be the number of nodes inside the support domain and $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be the field nodes. Then, the shape functions $\phi_1(\boldsymbol{x}), \ldots, \phi_n(\boldsymbol{x})$ corresponding to the nodes $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ are obtained as the solution of the linear system,

$$G\left[\phi_1(\boldsymbol{x}), \ldots, \phi_n(\boldsymbol{x}), \phi_{n+1}(\boldsymbol{x}), \ldots, \phi_{n+m}(\boldsymbol{x})\right]^{\mathrm{T}} = \left[b_1(\boldsymbol{x}), \ldots, b_n(\boldsymbol{x}), p_1(\boldsymbol{x}), \ldots, p_m(\boldsymbol{x})\right]^{\mathrm{T}}, \tag{5}$$



(a) RPIM shape function $\phi_i(\boldsymbol{x})$ (b) modified RPIM shape function $\phi_i(\boldsymbol{x})$

Figure 3: Example of RPIM, and modified RPIM shape functions corresponding to the node $\boldsymbol{x}_i = (0.8, 0.7)$, where $\alpha_x = \alpha_y = 4.0$, and the number of divisions of the domain Ω is 5×5, as shown in Fig. 4.

where $b_i(\boldsymbol{x})$ are radial basis functions (RBF [4, 7]) centred at the field nodes, $\{p_1(\boldsymbol{x}), p_2(\boldsymbol{x}), \ldots\}$ are monomial terms such as $\{1, x, y, \ldots\}$ and G is a symmetric matrix defined as

$$G = \begin{bmatrix} B_0 & P_0 \\ P_0^{\mathrm{T}} & 0 \end{bmatrix}, \quad B_0 = \begin{bmatrix} b_1(\boldsymbol{x}_1) & \cdots & b_n(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ b_1(\boldsymbol{x}_n) & \cdots & b_n(\boldsymbol{x}_n) \end{bmatrix}, \quad P_0 = \begin{bmatrix} p_1(\boldsymbol{x}_1) & \cdots & p_m(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ p_1(\boldsymbol{x}_n) & \cdots & p_m(\boldsymbol{x}_n) \end{bmatrix}. \tag{6}$$

The partial derivatives of the shape functions can also be obtained using the same matrix G, defined in (6) as

$$G \left[\tfrac{\partial}{\partial x}\phi_1(\boldsymbol{x}), \ldots, \tfrac{\partial}{\partial x}\phi_n(\boldsymbol{x}), \tfrac{\partial}{\partial x}\phi_{n+1}(\boldsymbol{x}), \ldots, \tfrac{\partial}{\partial x}\phi_{n+m}(\boldsymbol{x})\right]^{\mathrm{T}} = \left[\tfrac{\partial}{\partial x}b_1(\boldsymbol{x}), \ldots, \tfrac{\partial}{\partial x}b_n(\boldsymbol{x}), \tfrac{\partial}{\partial x}p_1(\boldsymbol{x}), \ldots, \tfrac{\partial}{\partial x}p_m(\boldsymbol{x})\right]^{\mathrm{T}}, \tag{7}$$

$$G \left[\tfrac{\partial}{\partial y}\phi_1(\boldsymbol{x}), \ldots, \tfrac{\partial}{\partial y}\phi_n(\boldsymbol{x}), \tfrac{\partial}{\partial y}\phi_{n+1}(\boldsymbol{x}), \ldots, \tfrac{\partial}{\partial y}\phi_{n+m}(\boldsymbol{x})\right]^{\mathrm{T}} = \left[\tfrac{\partial}{\partial y}b_1(\boldsymbol{x}), \ldots, \tfrac{\partial}{\partial y}b_n(\boldsymbol{x}), \tfrac{\partial}{\partial y}p_1(\boldsymbol{x}), \ldots, \tfrac{\partial}{\partial y}p_m(\boldsymbol{x})\right]^{\mathrm{T}}. \tag{8}$$

There are several types of RBFs as introduced in [7]. One reasonable choice is a multi-quadrics function,

$$b_i(\boldsymbol{x}) = (\|\boldsymbol{x} - \boldsymbol{x}_i\|^2 + (\alpha_c d_c)^2)^q,$$

where α_c and q are shape parameters. It is known in this case that the values $\alpha_c = 4.0$ and $q = 1.03$ give a satisfactory result, and hence, we use these parameters throughout this paper. See [7] for more information. Note that the support domain depends on the point of interest $\boldsymbol{x}$, and therefore, the matrix G varies with the position of the point of interest, $\boldsymbol{x}$.

2.2 Modified definition of shape functions

We give another definition of the shape functions, such that the matrix G is independent of the position of $\boldsymbol{x}$ in a certain region. In order to construct shape functions satisfying the property, we

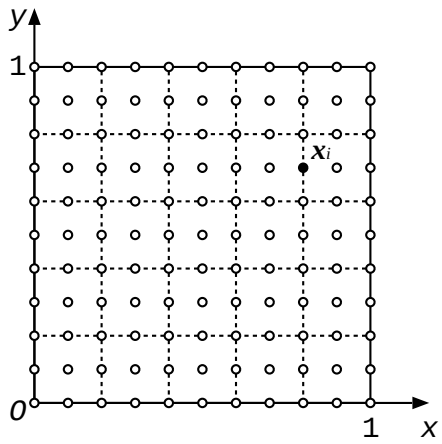


Figure 4: Example of nodes (11×11) in a problem domain $\Omega = [0, 1] \times [0, 1]$. Black dot indicates a node $\boldsymbol{x}_i = (0.8, 0.7)$. Domains divided by the dotted lines are examples of subdomains used for shape function evaluation.

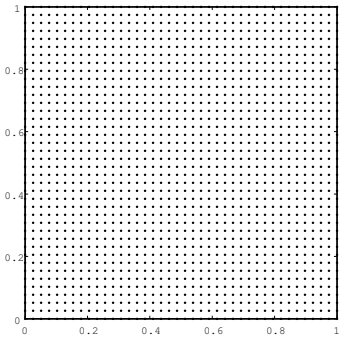
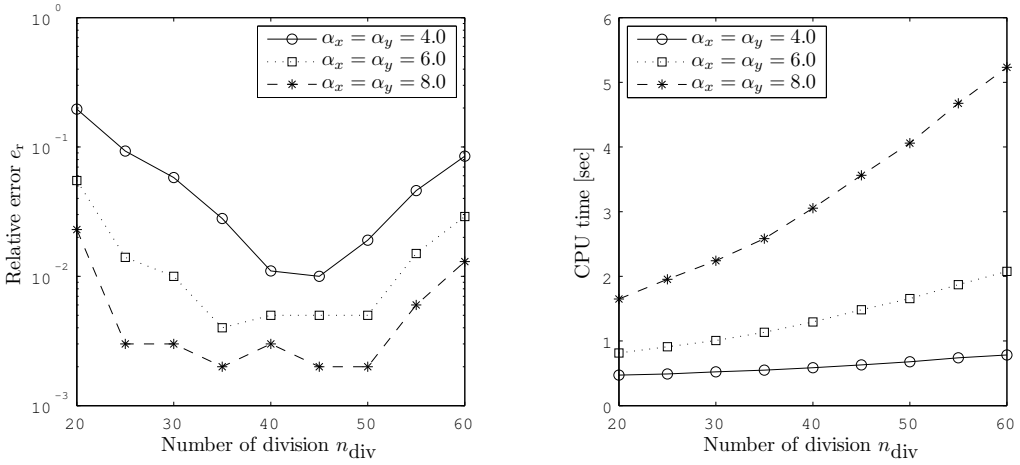


Figure 5: Domain ($[0, 1] \times [0, 1]$) and field nodes (40×40)

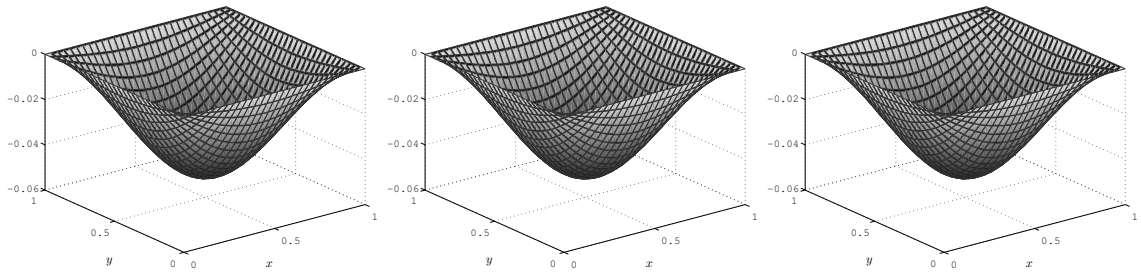
first consider a division of the problem domain. More specifically, we make rectangular subdomains $\Omega_1, \Omega_2, \ldots$ by dividing a rectangular region including the problem domain Ω (See Fig. 2). Let a support domain corresponding to a point of interest, $\boldsymbol{x}$, be a rectangular region centred at the centre of the corresponding subdomain. The size of the support domain, d_x and d_y, is determined as given in (4), using user-defined parameters, α_x and α_y. An example of the support domain is shown in Fig. 2. A subdomain is selected from the position of the point of interest $\boldsymbol{x}$, and the support domain is determined, not from the point of interest $\boldsymbol{x}$, but from the centre of the selected subdomain, Ω_i.

The shape functions corresponding to the new support domain can be defined using the same idea of the original RPIM as described in (5) and (6). We call these modified RPIM shape functions. An example of the modified RPIM shape function, compared to the conventional RPIM shape function, is shown in Fig. 3. Here, we assume that the problem domain, Ω, is $[0, 1] \times [0, 1]$, and the nodes are distributed in Ω uniformly and Ω is divided into 5×5 subdomains, $\Omega_1, \ldots, \Omega_{25}$, as shown in Fig. 4. In addition, we used $\alpha_x = \alpha_y = 4.0$ as the parameters, determining the support domain size.



(a) Relative errors (b) CPU times for solving PDE [sec]

Figure 6: Relative errors and CPU times as a function of number of divisions



(a) Exact solution of eq. (9) (b) Solution obtained by RPIM (Relative error $e_r = 3.09 \times 10^{-3}$) (c) Solution obtained by modified RPIM (Relative error $e_r = 2.51 \times 10^{-3}$)

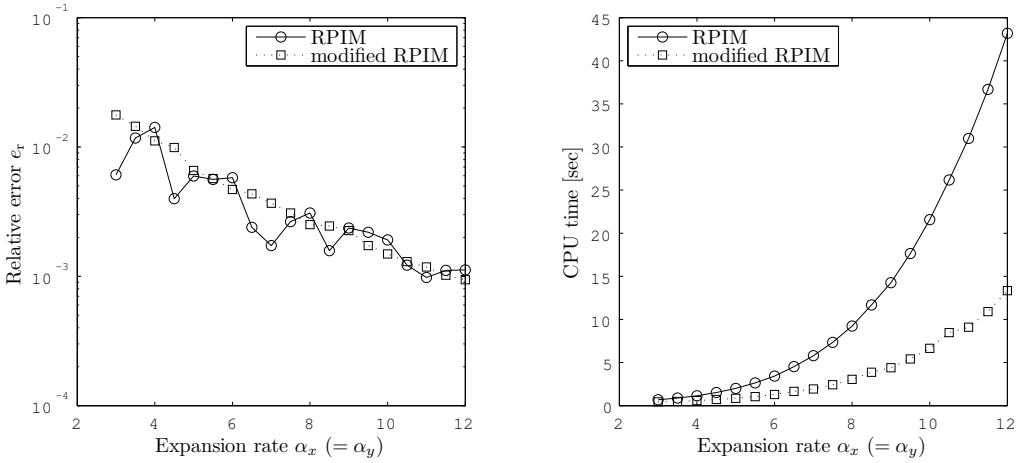
Figure 7: Exact solution and approximate solutions obtained by RPIM and modified RPIM

Some of our numerical experiments indicate that the modified RPIM shape function tends not to differ significantly from the RPIM shape function, although its definition is different (Fig. 3).

The matrix G for evaluation of the modified RPIM shape functions is constant in the corresponding subdomain, and thus, the matrix can be constructed in advance of the shape function evaluation process. Let n_s be the number of subdomains and $G_1, \ldots, G_{n_s}$ be matrices corresponding to the subdomains, $\Omega_1, \ldots, \Omega_{n_s}$: then, the matrices G_i are defined by (6), and the solution of the linear system (5) can be obtained efficiently by the following procedure:

1. (Pre-processing) Divide Ω into subdomains $\Omega_1, \ldots, \Omega_{n_s}$ and construct corresponding matrices $G_1, \ldots, G_{n_s}$. Then, compute a decomposed form of every matrix G_i using decomposition such as LU or block LDL^T [8]. Here, we assume that all matrices are decomposed as $G_i = L_i U_i$.

2. (Construction of the linear system) Construct the matrix A and the right-hand side vector $[y_1, \cdots, y_N]^T$ of the linear system (2) using a Gaussian integration rule. In this step, the value of the functions $\phi_i(\boldsymbol{x})$ and the corresponding gradients $\nabla\phi_i(\boldsymbol{x})$ at each integration point are required to compute (3) using the numerical integration. For an integration point, $\boldsymbol{x}$, the values $\phi_i(\boldsymbol{x})$ and $\nabla\phi_i(\boldsymbol{x})$ are obtained as follows:



(a) Relative errors (b) CPU times [sec]

Figure 8: CPU times required for construction of linear system (2) and relative errors. The number of subdomains n_s is fixed to 40×40 for modified RPIM.

(a) Find a subdomain, Ω_i, corresponding to $\boldsymbol{x}$ (See Fig. 2(b))

(b) Solve the linear systems (5), (7) and (8). The solutions of these linear systems can be obtained with low computational cost using the decomposed form obtained in pre-processing.

3 Numerical Examples

As a representative example, we consider the simple boundary value problem,

$$\begin{cases} -\Delta u = -\sin \pi x \sin \pi y & \text{in} \quad \Omega = [0,1] \times [0,1], \\ u = 0 & \text{on} \quad \Gamma. \end{cases} \tag{9}$$

The exact solution is $-\frac{1}{2\pi^2} \sin \pi x \sin \pi y$. We use 40×40 uniformly distributed field nodes in Ω throughout this numerical test, as shown in Fig. 5. Thus, both nodal spacing variables, d_{cx} and d_{cy}, are 1/39. The number of the background cells for numerical integration required in (3) is 30×30, and 3×3 Gaussian integration points are adopted at each background cell. All experiments in this section were conducted with an Intel Pentium 4 processor 3.0 GHz, Linux operating system, g++ 3.3.2 compiler and 1 GB memory.

The modified RPIM requires a new parameter, n_s (number of subdomains). This parameter plays an important role in improving the computational performance. The relative errors of approximate solutions and the CPU times required to obtain approximate solutions as a function of number of divisions, d_{div}, are shown in Fig. 6(a) and (b), respectively. Here, the number of divisions, d_{div}, means that the domain, Ω, is divided into $d_s = d_{\text{div}} \times d_{\text{div}}$ subdomains. Note that the CPU time results include pre-processing, i.e. construction of matrices G_i and their decompositions.

The definition of the relative error e_r is

$$e_r = \|u_{\text{exact}}(\boldsymbol{x}) - u_{\text{num}}(\boldsymbol{x})\|_\infty / \|u_{\text{exact}}(\boldsymbol{x})\|_\infty,$$

where $u_{\text{exact}}(\boldsymbol{x})$ is the exact solution, and $u_{\text{num}}(\boldsymbol{x})$ is a numerical solution. From the results of the relative errors, we see that $n_{\text{div}} = 40$ gives a reasonable solution for $\alpha_x = \alpha_y = 4.0$, and the optimal

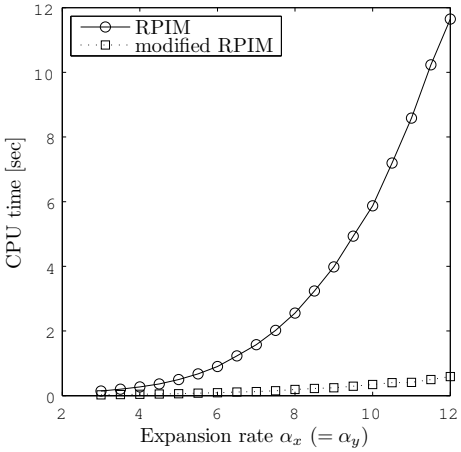


Figure 9: CPU times for sampling the solution of PDE (50×50 sampling points)

value of n_{div} is almost independent of the size of the support domain in this case. The results shown in Fig. 6(b) illustrate that the CPU time increases as the number of divisions increases. This is a natural sequel since the number of divisions determines the number of the matrices G_i.

These results indicate that 40×40 subdomains provide a reliable solution with reasonable cost in this case. Fig. 7 illustrates a comparison of the solutions obtained by RPIM and modified RPIM for $\alpha_x = \alpha_y = 8.0$. No major difference exists between these two approximate solutions.

The other parameters are α_x and α_y, which determine the size of the support domain. Fig. 8(a) and (b) illustrates the relative errors and the CPU time required for the construction of the linear system (2) as functions of α_x.

Here, we used 40×40 subdomains which is selected according to the results in Fig. 6(a), we applied LU for the decomposition of G_i for solving (2), and $\alpha_x = \alpha_y$. In the case of modified RPIM, the CPU time shown in this figure includes pre-processing, construction of the linear system (2), and the system solving process.

The results of the relative errors (Fig. 8(a)) show that the accuracy of the approximate solutions obtained by modified RPIM decreases with an increase in the size of the support domain, and behaves similarly to the conventional RPIM. The CPU time results shown in Fig. 8(b) indicate that the numerical solutions of the modified RPIM can be obtained with lower computational cost and the difference increases with α_x. support domain. In the case of $\alpha_x = \alpha_y = 8.0$, for example, the CPU time required for the construction of the linear system (2) is about 3.05 sec for the modified RPIM—about 1/3 of the CPU time of the RPIM (9.22 sec).

Sampling of approximate solutions can be performed at any point in Ω by substituting sampling points into (1). The sampling process is also accelerated in the modified RPIM because the shape functions in (1) can be evaluated using the decompositions of G_i obtained in the pre-processing. The CPU times required for sampling approximate solutions are shown in Fig. 9, where 50×50 grid points are used for the sampling. The CPU times in this figure does not include pre-processing in the case of modified RPIM. For example, in the case of $\alpha_x = \alpha_y = 8.0$, the CPU time required for the sampling using the modified RPIM is 0.19 sec—about 1/13 of the CPU time for the RPIM (2.55 sec).

4 Conclusion

The present work has developed an efficient algorithm for meshfree method based on radial point interpolation. In the method, the shape functions have been defined depending on subdomains given as a division of a bounding rectangle of the problem domain. The new definition allows us fast evaluation of the shape functions since the matrices for shape function evaluation and their decomposed forms can be calculated in pre-processing in advance of the evaluation process. Our numerical results indicate that the new definition of the shape functions provides reliable solutions at low computational cost. In addition, the results indicate that the new definition contributes to fast evaluation of approximate solutions.

Acknowledgement

This research was supported by Grant-in-Aid for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan (No. 18760067) and Grant-in-Aid for the 21st Century COE "Center of Excellence for Disaster Mitigation of Urban Cultural Heritage" from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] T. Belytschko, Y. Y. Lu and L. Gu, Element-Free Galerkin Methods, International Journal for Numerical Methods in Engineering, Vol. 37, Issue 2, 229–256 (1994).

[2] P. Lancaster and K. Salkauskas, Surfaces Generated by Moving Least Squares Methods, Mathematics of Computation, Vol. 37, 141–158 (1981).

[3] G. R. Liu, Mesh free methods: moving beyond the finite element method, CRC Press, Boca Raton (2002).

[4] J. G. Wang and G. R. Liu, A point interpolation meshless method based on radial basis functions, International Journal for Numerical Methods in Engineering, Vol. 54, Issue 11, 1623–1648 (2002).

[5] K. M. Liew and X. L. Chen, Mesh-free radial point interpolation method for the buckling analysis of Mindlin plates subjected to in-plane point loads, International Journal for Numerical Methods in Engineering, Vol. 60, Issue 11, 1861–1877 (2004).

[6] G. R. Liu, G. Y. Zhang, Y. T. Gu and Y. Y. Wang, A meshfree radial point interpolation method (RPIM) for three-dimensional solids, Computational Mechanics, Vol. 36, No. 6, 421–430 (2005).

[7] G. R. Liu and Y. T. Gu, An introduction to meshfree methods and their programming, Springer, Dordrecht (2005).

[8] G. H. Golub and C. F. Van Loan, Matrix computations 3rd edition, The Johns Hopkins University Press, Baltimore (1996).