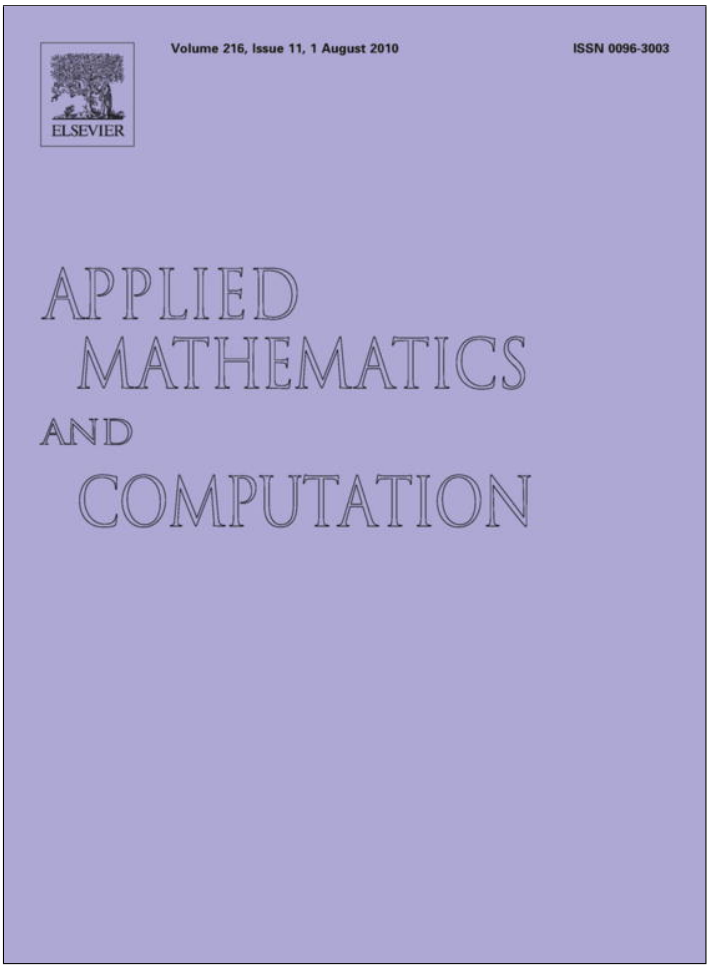

Applied Mathematics and Computation 216 (2010) 3355–3364

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A parallel implementation of overlapping Schwarz method for the dual reciprocity method

Ke Chen [a], Kamal Shanazari [b,*]

[a] Department of Mathematical Sciences, The University of Liverpool, Liverpool L69 7ZL, UK
[b] Department of Mathematics, University of Kurdistan, Sanandaj, Iran

ARTICLE INFO

Keywords:
Boundary element method
Overlapping domain decomposition
Ill-conditioning
Parallel computing
Partial differential equations

ABSTRACT

We present a parallel algorithm for the overlapping domain decomposition boundary integral equation method for two dimensional partial differential equations. In addition to the improvement of the ill-conditioning and the computational efficiency achieved by domain partitioning, using a parallel computer with p processors can offer up to p times efficiency. Assuming direct solution is used throughout, partitioning the domain into p subregions and employing a processor for each subproblem, overall, result in p^2 times efficiency over using a single domain and a single processor, taking into account that a sequential algorithm of the underlying method can improve the computational efficiency at least p times over using a single domain. Some numerical results showing the efficiency of the parallel technique will be presented.

1. Introduction

The dual reciprocity method (DRM) is a generalized boundary element method (BEM) to solve a class of non-homogeneous partial differential equations (PDEs) which do not have an explicit fundamental solution. This method, first presented by Nardini and Brebbia [12], tackles the domain integrals, arising from the inhomogeneous term in the classical BEM, by approximating a particular solution [8] and applying the dual integration. The particular solution is approximated by using a class of well known functions called radial basis functions (RBFs).

There exist a large class of interpolating RBFs [11,16] that can be used by DRM. These include the linear $1 + r$, the polynomial $P_k(r)$, the thin plate spline (TPS) $r^2 \log r$, the Gaussian $\exp(-r^2/\beta^2)$ and the multiquadrics $\sqrt{\beta^2 + r^2}$ (with β a constant parameter). We shall mainly use a polynomial RBF, $1 + r^3$, here.

The RBFs are globally defined functions which result in dense interpolation matrices whose condition number can rapidly increase, especially for the large size problems. Another difficulty concerns the computational efficiency due to the dense coefficient matrices. To overcome the ill-conditioning some sort of localization such as compactly supported RBFs (CS-RBFs) [23] and domain decomposition method (DDM) [3] have been proposed. Using CS-RBFs results in sparse interpolation matrices which improve the conditioning and computational efficiency of the method.

In the DDM the domain is divided into a number of subdomains and the numerical method is applied to each subregion to obtain the solution for subproblems followed by assembling the global solution. Consequently, the final linear system formed by the numerical method is split into some small systems and the condition number is considerably reduced. This also increases the computational efficiency due to dealing with smaller matrices.

* Corresponding author.
E-mail addresses: k.chen@liverpool.ac.uk (K. Chen), k.shanazari@uok.ac.ir, kshanaz@liverpool.ac.uk (K. Shanazari).
URL: http://www.liv.ac.uk/~cmchenke (K. Chen).

0096-3003/$ - see front matter © 2010 Published by Elsevier Inc.
doi:10.1016/j.amc.2010.04.063

The DDM fall into two main categories: overlapping and non-overlapping variants as widely studied in the numerical solution of PDEs [21]. Popov and Power [14] applied the DDM to DRM in the non-overlapping variant. They showed that the DRM can be improved by using DDM. In particular, for the convection diffusion equations, they gained better accuracy when the original domain was partitioned into two and three subdomains. Their idea of using DDM was motivated by the work of Kansa and Carlson [10] on the RBFs for which they indicated that the best accuracy was achieved when the original domain was split into a number of *non-overlapping* subdomains. Following the work of Popov and Power the application of DDM in the DRM has been developed for many applications [17,5,7,6,18,15].

While the above work was based on non-iterative and non-overlapping method, recently, the accuracy of the DRM has been improved by applying the *overlapping* Schwarz method [19]. Although this approach enjoys the improvement of the ill-conditioning and computational efficiency due to the reduction of the matrices size, using a parallel computer can offer more improvement in the efficiency. This is the purpose of this paper to present a parallelization scheme of the DRM–Schwarz method presented in [19].

The use of multiple processors has now become a popular technique in many numerical methods with long execution time [2,22]. Much effort has been made to design parallel algorithms adaptive with these computers [1]. For some of the numerical methods, parallelization is either complicated or inefficient due to the large amount of communication between the processors which increases the running time. Instead, methods such as domain decomposition are naturally suited for parallel computing and this has increased the popularity of the DDM. The parallel implementation of the domain decomposition boundary element method has been carried out by some researchers (see [4,20]).

The alternating Schwarz method based on Jacobi is suitable for parallel computing. Here we present a parallel implementation of this method. Due to the parallel nature of the DDM, the simplest and most convenient implementation will be the case in which each processor deals with one subproblem. In this case the main part of the algorithm will be the communication between the processors for updating the artificial boundaries which is presented in this paper. Since there is no more communication in the algorithm, the time consumed is not considerable. This will be demonstrated by observing the CPU time for some cases. As will be shown, partitioning the domain into p subdomains and employing p processors for the underlying method, lead to p^2 times or even more efficiency over using a single domain. That is due to the fact that the method takes advantage of the use of both DDM and multi-processor. The contribution of the parallel computation to the improvement achieved will be evaluated and proved to be effective.

The rest of this paper will be organized as follows. The DRM is briefly described in Section 2. In Section 3 the alternating Schwarz method in the case of two and many subdomains will be reviewed. A parallelization scheme of the Schwarz method will be discussed in Section 4 and finally some numerical results together with the complexity analysis will be presented in Section 5.

2. The dual reciprocity method

We describe the method for the following Poisson equation,

$$\nabla^2(u) = b, \text{ in } \Omega, \tag{1}$$

where u is the dependent variable, Ω is the domain and b is the non-homogeneous part, which can generally be in the form,

$$b = f\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right).$$

The boundary condition can be of the Dirichlet type ($u = \bar{u}$ on Γ_D) or Neumann type ($\frac{\partial u}{\partial n} = \bar{q}$ on Γ_N) or Robin type (a combination of both), where $\Gamma_D \bigcup \Gamma_N = \Gamma$, is the boundary enclosing Ω. Applying the weighted residual technique and using Green's theorem, the integral formulation for Eq. (1) is given by

$$c_i u_i = \int_\Gamma \left(u \frac{\partial u^*}{\partial n} - u^* \frac{\partial u}{\partial n}\right) d\Gamma + \int_\Omega u^* b \, d\Omega, \tag{2}$$

where c_i is a coefficient depending on the location of the point i and u^* is the fundamental solution given by

$$u^* = -\frac{1}{2\pi} \ln r,$$

where r is the distance between the source point i and any field point under consideration. The above integral equation contains a domain integral corresponding to the non-homogeneous term, b. The main idea in the DRM is converting this domain integral to an equivalent boundary integral by approximating the function b in terms of interpolating functions at a certain number of boundary N and internal L nodes, that is,

$$b = \sum_{k=1}^{N+L} \alpha_k \phi_k, \tag{3}$$

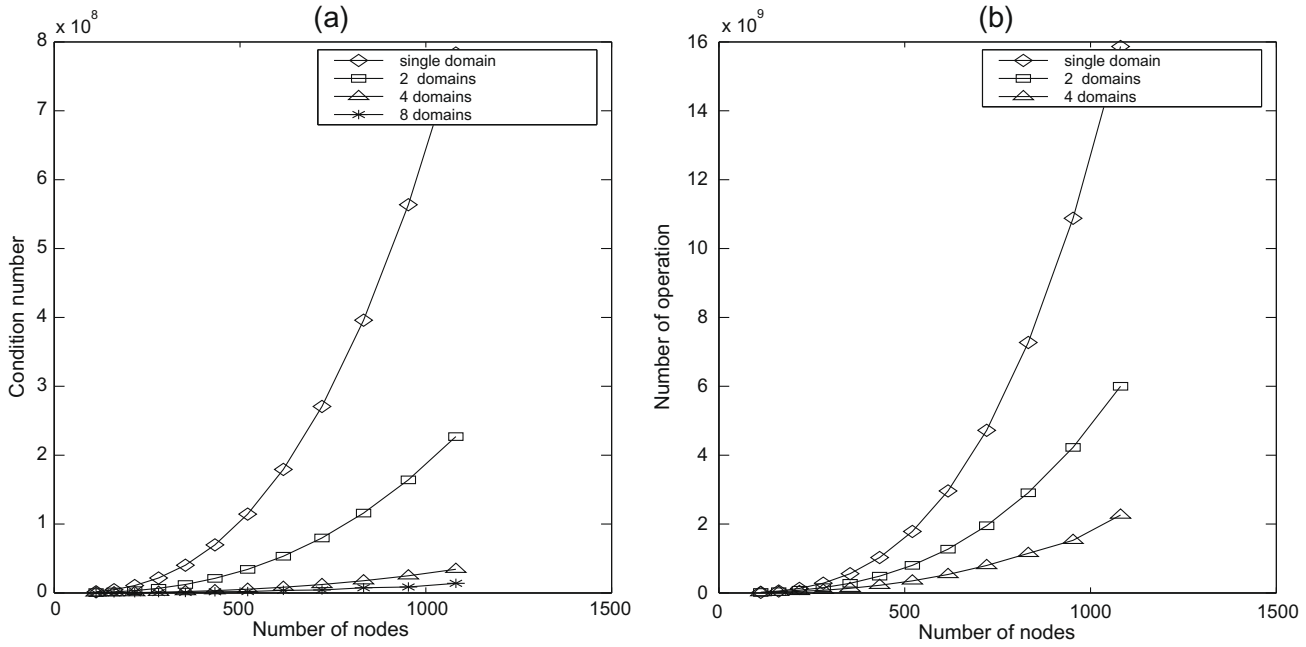

Fig. 1. The condition number of the interpolation matrix and the DRM cost are shown respectively in (a) and (b) for various number of overlapping subdomains for a rectangular domain.

where ϕ_k are the RBFs and α_k are the corresponding coefficients. Now the functions $\hat{u}_k$ can be found such that

$$\nabla^2 \hat{u}_k = \phi_k.$$

With this approximation for b, the domain integral in Eq. (2) is converted to an equivalent boundary integral by applying the same idea as for the main equation, that is:

$$\int_\Omega b u^* \, d\Omega = \sum_{k=1}^{N+L} \alpha_k \int_\Omega (\nabla^2 \hat{u}_k) u^* \, d\Omega = \sum_{k=1}^{N+L} \alpha_k \left\{ \left(u^* \frac{\partial \hat{u}_k}{\partial n} - \hat{u}_k \frac{\partial u^*}{\partial n} \right) d\Gamma - \hat{u}_{ik} \right\}, \tag{4}$$

where $\hat{u}_{ik}$ is the known value of $\hat{u}_k$ at the source point i. Substituting (4) in (2) leads to the system of equations

$$Hu - Gq = (H\widehat{U} - G\widehat{Q})F^{-1}b, \tag{5}$$

where H and G are the BEM matrices, $\widehat{U}$ and $\widehat{Q}$ are matrices relating to $\hat{u}_k$ and their normal derivatives and F is the interpolation matrix relevant to the approximation in (3). Also u and q are the vectors containing the potentials and the normal derivatives and b is a vector of the values of the right side of Eq. (1). To find more details about the DRM, see [13]. There are at least two points in motivating the use of DDM in the DRM. Firstly, the accuracy of the RBF interpolation depends upon the condition number of the interpolation matrix. The condition number is sensitive to the size of this matrix and it can be considerably reduced by domain partitioning due to the small sized matrices, as shown in Fig. 1a. Secondly, the DRM formulation contains some operations on full matrices which are costly. A cost analysis for the DRM containing the number of arithmetic operations to construct the linear system of equations in (5) and solving it by a direct method indicate that although apparently some extra points arising from the overlap regions are involved in calculations, the overall operations indicates a considerable reduction in comparison with the case of using a single domain, as depicted in Fig. 1b.

3. An overlapping Schwarz method

We first describe an overlapping Schwarz method for PDEs using two subdomains and then generalize it to multi-domain DDM. We explain the method for the following Poisson equation with a Dirichlet boundary condition,

$$\nabla^2(u) = b, \text{ in } \Omega, \quad u = \bar{u}, \text{ on } \Gamma. \tag{6}$$

Other kinds of boundary conditions can be considered similarly. Let the domain Ω be bounded by Γ be divided into two subdomains Ω_1 and Ω_2 bounded by $\partial\Omega_1$ and $\partial\Omega_2$ respectively (Fig. 2). The artificial boundaries for these subdomains are denoted by Γ_1 and Γ_2; consequently $\partial\Omega_i \backslash \Gamma_i$ is the part of the boundary of the subdomain Ω_i which is not interior to Ω. Also let $\overline{\Omega} = \Omega \cup \partial\Omega$ denotes the closure of the domain. In order to describe the method, suppose that u_i^n stands for the approx-

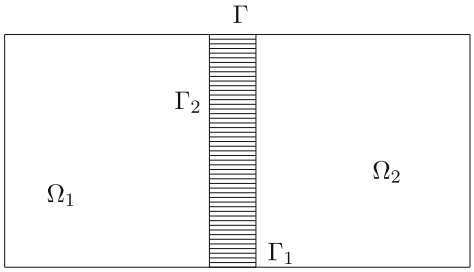

Fig. 2. The domain Ω has been split into subdomains Ω_1 and Ω_2.

imate solution on $\overline{\Omega}_i$ after n iterations, $u_1^n|_{\Gamma_2}$ is the restriction of u_1^n to Γ_2 and $u_2^n|_{\Gamma_1}$ is the restriction of u_2^n to Γ_1. In this method an initial guess u_2^0, for the values of u in Ω_2, is required to start the iteration. In fact only the values along the Γ_1 is necessary. The Schwarz method is then proceeded with the following iteration to obtain the solution,

$$\begin{cases} \nabla^2 u = b & \text{in } \Omega_1, \\ u_1^n = \bar{u} & \text{on } \partial\Omega_1 \setminus \Gamma_1, \\ u_1^n = u_2^{n-1}|_{\Gamma_1} & \text{on } \Gamma_1, \end{cases} \quad \text{and} \quad \begin{cases} \nabla^2 u = b & \text{in } \Omega_2, \\ u_2^n = \bar{u} & \text{on } \partial\Omega_2 \setminus \Gamma_2, \\ u_2^n = u_1^n|_{\Gamma_2} & \text{on } \Gamma_2. \end{cases} \tag{7}$$

The above iteration is performed until a desirable accuracy is achieved. More precisely for a given positive value, ϵ, the inequalities $|u_1^n - u_1^{n-1}| < \epsilon$ and $|u_2^n - u_2^{n-1}| < \epsilon$ are satisfied. It should be noted that the values $u_2^n|_{\Gamma_1}$ and $u_1^n|_{\Gamma_2}$ are approximated by interpolating in Ω_2 and Ω_1 respectively. The RBF interpolation is used for this purpose and the same RBF as in (3) is employed in this work, that is $1 + r^3$. To find more details on alternative approaches, see [21]. The formulation of the iterative scheme in (7) when applied to the DRM has been presented in [19] and proved that it is equivalent to the block Gauss–Seidel method. The alternative Schwarz method can be constructed based on the block Jacobi. In this case the boundary conditions for the subdomains are updated together and each subproblem is solved independently. This method, of course, reduces the rate of convergence but is suitable for parallel computing.

3.1. A multi-domain algorithm

Extending the alternating Schwarz method to the case of many subdomains is straightforward. Since in the present work we are interested in parallel algorithms, here we present a parallel algorithm based on the Jacobi method. Let the domain Ω be partitioned into p overlapping subdomains, then the Jacobi algorithm can be presented as follows,

Algorithm 1. An alternating Schwarz algorithm based on Jacobi

1. Update the boundary condition for all the subdomains.
2. Apply the DRM to each subproblem.
3. If the desirable accuracy has been achieved stop otherwise go to step 1.

In the first iteration, the boundary conditions are determined using an initial guess for all the subdomains and for the next iterations the solution from the previous iteration is used to update the boundary conditions. The method is suitable for parallel implementation since each subproblem can be solved independently.

4. Parallelization of the Schwarz method for a rectangular domain

In this section we present a parallelization scheme for the Jacobi–Schwarz method described in Algorithm 1. We consider a rectangular domain divided into p rectangular overlapping subdomains as depicted in Fig. 3a. The subdomains are numbered from *1* to p and the subproblem i, $i = 1, 2, \ldots, p$ is dealt with by processor i. In order for the subdomains to identify their neighbours and therefore to facilitate the communications between the processors, we use a special way of numbering the subdomains. Let N_x and N_y be the number of the subdomains in each row and column respectively and D_{ij} represent the subdomain located in row i and column j. We number the subdomain D_{ij} and thus its processor by

$$p(D_{ij}) = (i-1)N_x + j, \quad i = 1, \ldots, N_y, \; j = 1, \ldots, N_x,$$

which produce the numbers of Fig. 3a for a particular case of nine subdomains. As shown in Fig. 3a, each subproblem couples with at least two and at most four subproblems. For example subproblem 1 couples with 2 and 4 and subproblem 5 will be communicating with 4, 2, 6 and 8. Having identified the processors corresponding to the neighbours, each processor sends its artificial boundaries coordinates to the relevant processors and returns the updated boundary values from the same processors after solving the subproblems in each iteration. In order to make a reliable communication between the processors, two arrays $S(1:4)$ and $R(1:4)$ are employed for each processor as follows,

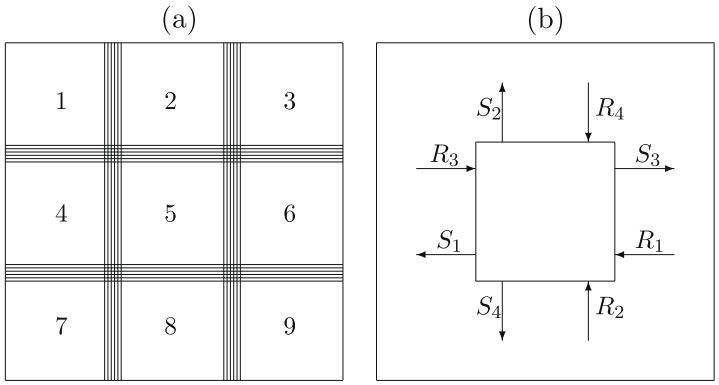

Fig. 3. A rectangle divided into nine subdomains are displayed in (a) and (b) shows how the arrays S and R control the communications for a processor.

$$\begin{cases} S_1 = ID - 1, & j \neq 1, \\ S_2 = ID - N_x & i \neq 1, \\ S_3 = ID + 1 & j \neq N_x, \\ S_4 = ID + N_x & i \neq N_y, \\ S_k = 0 & \text{otherwise}, \end{cases} \quad \text{and} \quad \begin{cases} R_1 = ID + 1, & j \neq N_x, \\ R_2 = ID + N_x & i \neq N_y, \\ R_3 = ID - 1 & j \neq 1, \\ R_4 = ID - N_x & i \neq 1, \\ R_k = 0 & \text{otherwise}. \end{cases} \tag{8}$$

S will be assigned the identification number of processors whose corresponding subdomains are located, respectively, in the left, up, right and down the side of the current subdomain which receive/send the information from/to the current processor with the identification, ID, i.e. the left side of (8) (see Fig. 3b). For processor k, if $S_k = 0$, it means that the receiver/sender processor corresponding to k does not exist. For example, for processor *1*, $S_1 = 0$ which means that there is no communication to the left side of subdomain *1*. The array R is used to store the *ID* number of the processors which send/receive the information to/from the current processor in response to the message controlled by S. This will be in the same order as in S but in the opposite direction, i.e. right, down, left and up respectively (see Fig. 3b). More precisely, the message which is sent/received to/from the left neighbour is received/sent from/to the right side and so on. This will set up the array R as indicated in the right side of (8). Within this agreement, the messages which are sent/received to/from S_i, $i = 1,2,3,4$, provided that $S_i > 0$, will be received/sent by R_i of the processor S_i and therefore a common *Tag*, say *Tag* = i, will be used for communication between S_i and R_i. This ensures that messages received from different directions will be identified by different Tags (see Fig. 4). As illustrated in Fig. 4, processor *5* sends its messages to processors $S_1 = 4$, $S_2 = 2$, $S_3 = 6$ and $S_4 = 8$ and these processors receive the messages from the processors, whose ID are available in their, R_1, R_2, R_3 and R_4, respectively, all of which represent 5. More specifically, processor 5 sends its message to $S_1 = 4$, using *Tag* = 1 and processor *4* receives the same message from processor $R_1 = 5$ via the same *Tag* and vice versa. Corresponding to the arrays S and R, two 2D arrays $X_S(1:n_b, 1:4)$ and $X_R(1:n_b, 1:4)$ are employed to store the coordinates of the artificial boundaries, exactly in the same order as in S and R. For instance, the first columns of X_S will be the coordinates of the boundary of the current subproblem which is shared with the left neighbour and will be sent to this neighbour and the first column of X_R will be the coordinates of the boundary of the right neighbour which is shared with the current subproblem and will be received from this neighbour and so on. The variable n_b represents the global maximum number of the boundary nodes which is sent in each communication. Similarly, two 2D arrays $U_S(1:n_b, 1:4)$ and $U_R(1:n_b, 1:4)$ are used for storing the artificial boundary values exactly in the same order as in X_S and X_R.

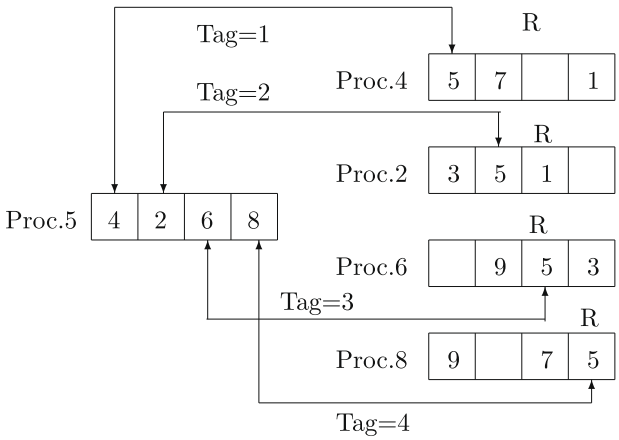

Fig. 4. Processor 5 sends its messages to processors S_i, $i = 1,2,3,4$ and these processors receive the messages from processor 5, whose ID is stored in R_i of processor S_i.

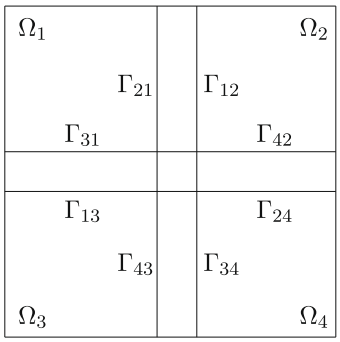

Fig. 5. The domain has been partitioned into four subdomains each of which couples with two subdomains.

4.1. The parallel algorithm

In this section we present a parallelization scheme of Algorithm 1 described in Section 3. Having produced an uniform mesh in each subdomain by the corresponding processor, the coordinates of the artificial boundaries are sent to the appropriate processors in the first communication. This stage is performed only once in order for the processors to identify the boundaries of their neighbours for the updating stage. The other communication will be for updating the artificial boundaries which is performed iteratively. We summarize the parallel algorithm for a general number of processors and present a simple illustration for a problem with four subdomains and four processors in Fig. 6. As shown in Fig. 5, the artificial boundary of subproblem i, denoted by Γ_i, is a combination of Γ_{ij} and Γ_{ik} which are located in the subdomains j and k respectively. Below for each step of the algorithm see the corresponding step of the illustration in Fig. 6.

Algorithm 2. A parallel scheme of Schwarz–Jacobi for the DRM

1. The mesh is produced, the real boundary conditions are set up and the artificial boundaries are initialized. Note that $\bar{u}$ is a known function which represents the solution function on the real boundary.
2. (I) The coordinates of the artificial boundaries of the current processor stored in $X_S(:,i)$, $i = 1,2,3,4$, are sent to processors S_i, for the case of $S_i > 0$.
   (II) The coordinates of the artificial boundaries of the neighbours are received from processors R_i, $i = 1,2,3,4$, for the case of $R_i > 0$ and will be stored in $X_R(:,i)$.
   For $n = 1,2,\ldots$ (until obtaining the desirable accuracy).
3. The linear system in (5) is constructed and solved to obtain the solution u^n (for subproblem i, $f_i = (H_i\widehat{U_i} - G_i\widehat{Q_i})F_i^{-1}b_i$).
4. The artificial boundaries are updated using the most recent solution and an interpolation operator.
5. (I) The most recent solution of the artificial boundaries of the neighbours, stored in $u_R(:,i)$, $i = 1,2,3,4$, are sent to processors R_i, for the case of $R_i > 0$.
   (II) The most recent solution of the artificial boundaries of the current processor are received from processors S_i, $i = 1,2,3,4$, for the case of $S_i > 0$ and will be stored in $u_S(:,i)$.

5. Numerical results

In order to show the computational efficiency we consider two PDEs and solve them by Algorithm 2, using Fortran 77 and MPI directives [9]. In each example, the domain is divided into various numbers of subdomains and the DRM is applied to each subproblem by a processor. Therefore, the number of subdomains will be equal to the number of processors used.

The numerical solutions are obtained when the elements are approximated by linear functions and the collocation points are selected inside the elements, that is, discontinuous elements. The solution error is measured at the collocation points in L_2-norm. In each example, the results of the DDM are compared with the case of single domain. The total number of points in the subdomains including the overlap is equal to the number of points in the single domain.

Example 1. In this example an elliptic equation

$$\nabla^2 u + y\frac{\partial u}{\partial x} + x\frac{\partial u}{\partial y} + xyu = d,$$

with a Dirichlet boundary condition is solved in the rectangle $\{(x,y), 0 \leqslant x \leqslant 1, 0 \leqslant y \leqslant 1\}$. Here d has been selected such that the exact solution is

$$u(x,y) = 0.5 - 0.5\tanh(-4 + (4x-2)^2 + (4y-2)^2).$$

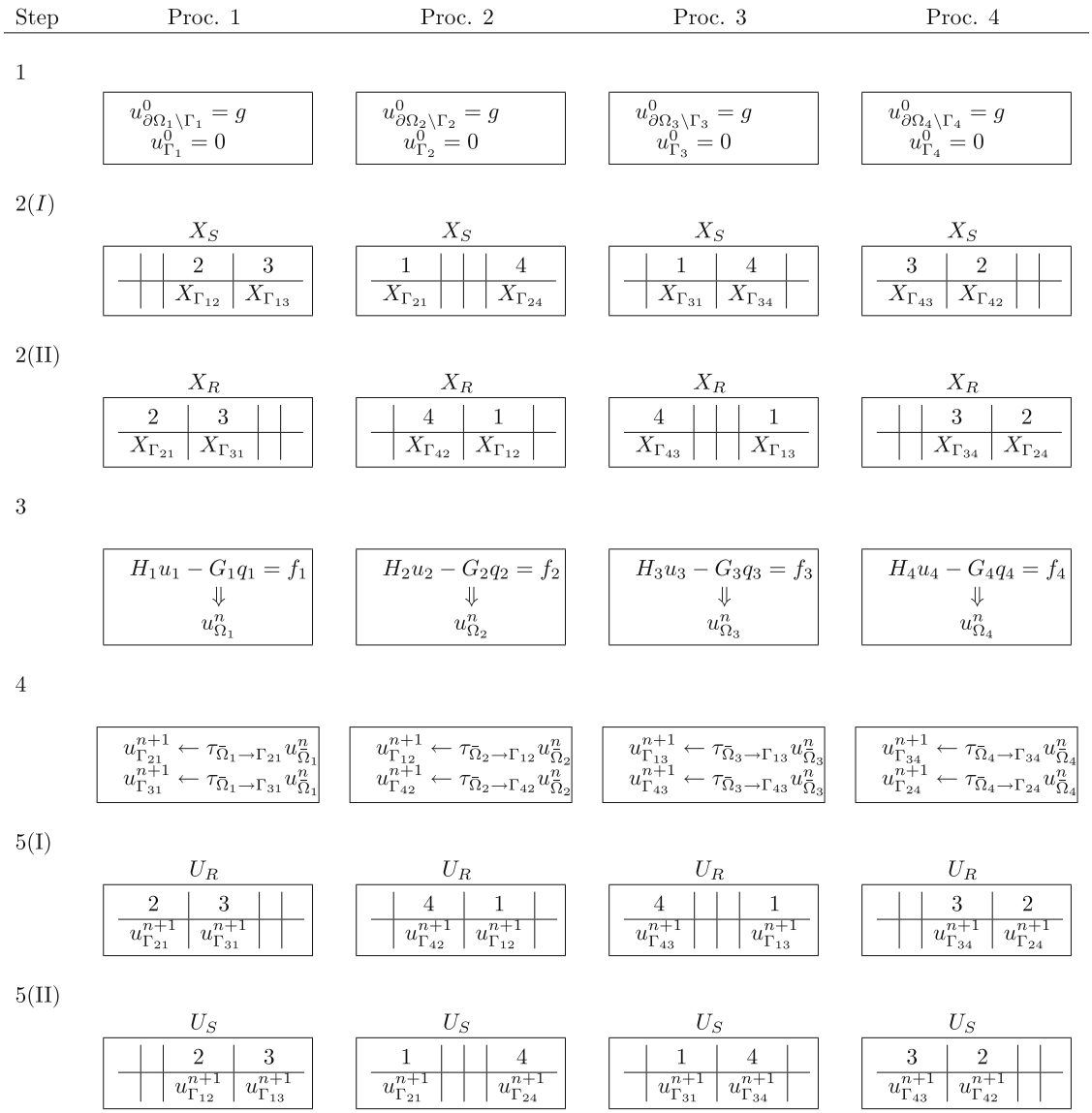

Fig. 6. The illustration of Algorithm 2; the numbers on the top of the boxes represent the ID number of the processors to/from which the messages are sent/received.

Table 1
The error values for potential function u for different numbers of subdomains with the number of iterations are listed for Examples 1 and 2.

Example 1, Number of nodes: 520			Example 2, Number of nodes: 1080		
Iteration	Subdomains	e_u	Iteration	Subdomains	e_u
	1	1.2E−2		1	1.0E−2
6	2	1.5E−2	5	2	4.7E−3
10	4	6.2E−3	6	3	2.7E−3
14	8	6.0E−3	7	4	6.0E−3

The error values for the interior points are presented in the case of using a single domain and different numbers of subdomains in Table 1. A significant improvement in the accuracy is observed in the DDM in comparison with the case of using a single domain, especially in the case of using four and eight subdomains. However, the more the number of subdomains is, the more the number of iterations is required to converge to the solution. This is a typical behaviour in the iterative DDM since the communication between the subproblems takes longer when more subdomains are involved.

Example 2. This example involves a convection diffusion equation

$$D\nabla^2 u - V_x \frac{\partial u}{\partial x} - ku = 0,$$

defined over an unit square where $u\,(\mathrm{kgm}^{-3})$ is the concentration of the transport of a substance, $V_x\,(\mathrm{ms}^{-1})$ is the velocity field in the x direction depending on the y coordinate, $V_x(y) = \frac{\lambda^2}{c_2}(y-0.5)^2$, with $\lambda = k - c_2^2$ and $c_2 = \log[c(1,0)/c(0,0)]$ and $k\,(\mathrm{s}^{-1})$ is the reaction constant. $u(0,y)$, $u(1,y)$, $\frac{\partial u}{\partial y}|_{y=0}$ and $\frac{\partial u}{\partial y}|_{y=1}$ are known as the boundary conditions and for the case $D = 1\ (\mathrm{m}^2\mathrm{s}^{-1})$ and $k = 50$, the exact solution for the above equation is given by $u = 300\exp\left[\frac{\lambda}{2}(y^2 - y)\right]\exp(c_2 x)$. Again the problem was solved in different cases as in Example 1. The L_2-norm of the relative error for the boundary and interior points are presented in Table 1. One can observe that the overall results using multi-domain are better than applying a single domain.

5.1. Analysis of complexity

As indicated in Fig. 1b, a sequential algorithm of the Schwarz method, applied to the DRM, itself, offers a considerable saving time in comparison with the case of single domain. It can be verified that the total number of operations in the DRM when using n nodes is evaluated by

$$F(n) = \frac{21}{2}n^3 + 68n^2 + 5n + 8. \tag{9}$$

Consequently, using p processors leads to the efficiency of

$$q(p,n) = \frac{F(n)}{pF(n_p)} \approx p, \tag{10}$$

over using a single domain where n is the total number of nodes, n_p is the number of nodes in each subproblem given by $n_p = n/p + overlap$ and thus $pF(n_p)$ is the total operations in the case of using p subdomains. Note that q in (10) is a function of both p and n and can be improved by increasing n and p, as indicated in Fig. 1b. As in the present parallel algorithm, each processor performs one subproblem, employing one processor turns the problem to the case of a single domain. Whereas employing $p > 1$ processors on one hand takes the advantage of the DDM to improve the CPU time up to q times, as indicated in (10), and on the other hand gains up to p times improvement due to the use of multi-processor. Therefore, in this case, overall $pq \approx p^2$ times efficiency is achieved over using a single domain. This distinguishes the present algorithm from many parallel algorithms in which the improvement achieved in CPU time by employing p processors is only up to p times.

We examined our parallel algorithm by Examples 1 and 2 and measured the CPU time for various numbers of nodes. The number of operations evaluated by (9) and the CPU time together with the overall and partial efficiency for different numbers of processors and nodes are listed in Table 2. One can observe the large improvement in the overall efficiency which depends on the size of the problem, i.e. the larger the problem is, the more improvement is achieved. For example, applying this algorithm with eight processors to problems with 350 and 950 nodes result in, respectively, 34.5 and 108 times saving in CPU (see Table 2). In the case of using eight processors with 520 points, this improvement is 61 times saving in CPU, i.e. nearly p^2 times improvement in computational efficiency. However, as highlighted before, a large part of this improvement is due to saving in arithmetic operations which is not a consequence of the parallel bit and can be even obtained by a sequential algorithm. In fact this is the part of the efficiency which depends on the size of the problem. This part of the improvement can be evaluated by (10) and is given in Table 2 denoted by sequential efficiency (q). The real contribution of the parallel-

Table 2
The flop counts and the CPU time together with the overall and partial efficiency.

Size n_p	Processors p	Cost $pF(n_p)$	Sequential efficiency (q) over $p = 1$	CPU (s)	Overall efficiency over $p = 1$	Partial efficiency over $p = 1$
350	1	5.36E9		57.60		
220	2	2.66E9	2.02	15.6	3.7	1.8
136	4	1.29E9	4.16	4.3	13.4	3.2
94	8	9.06E8	6.05	1.7	34.5	5.7
520	1	1.70E10		183		
315	2	7.60E9	2.23	43	4.3	1.9
188	4	3.33E9	5.11	10.0	18.3	3.5
125	8	2.03E9	8.37	3.0	61.0	7.3
720	1	4.47E10		485		
424	2	1.85E10	2.42	104	4.7	1.9
248	4	7.54E9	5.92	22	22.1	3.7
160	8	4.19E9	10.67	5.8	83.2	7.8
950	1	1.03E11		1160		
550	2	4.01E10	2.57	224	5.2	2.0
316	4	1.54E10	6.69	44.1	26.1	3.9
200	8	7.95E9	12.95	10.8	108.0	8.3

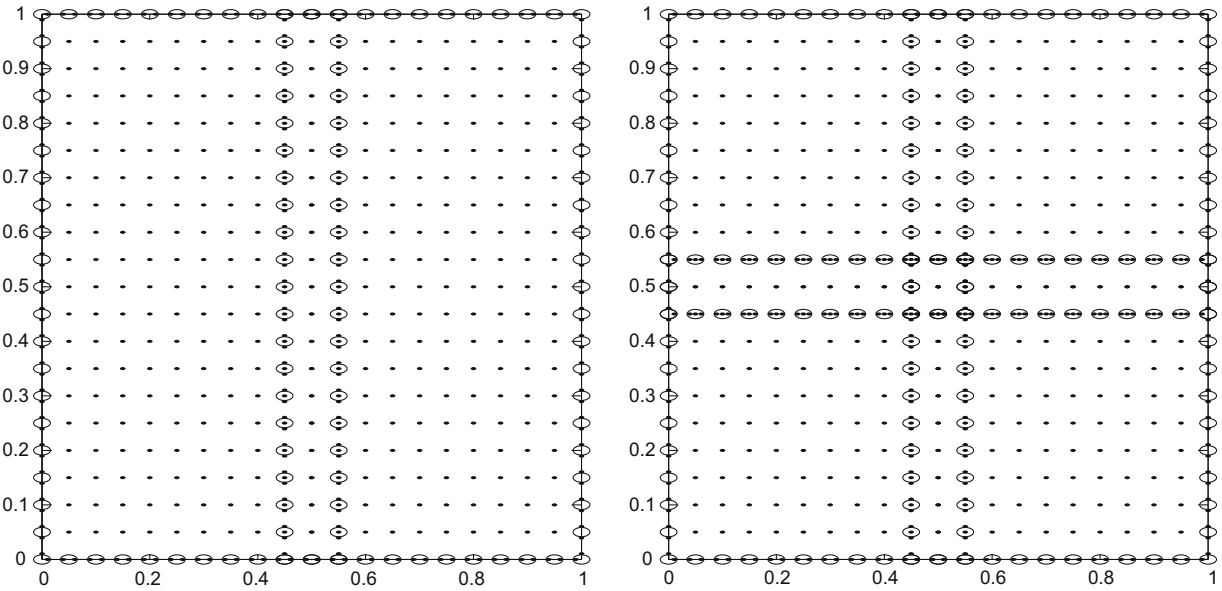

Fig. 7. The domain partitioning are displayed in the case of 2 and 4 subdomains with a small amount of overlap.

ization can be calculated by dividing the overall efficiency by the sequential efficiency, referred to as parallel efficiency in Table 2. As observed, the results meet our expectation and the improvement scales well with the number of the processors.

In our previous work [19], we have shown that in the DRM–DDM the size of the overlap affects the rate of convergence. Increasing the size of overlap, on one hand, increases the speed of convergence and, on the other hand, reduces the computational efficiency due to the size of the subproblems. In order to achieve more improvement in the efficiency, a small amount of overlap was used (see Fig. 7). This resulted in a reasonable speed of convergence, though the number of iterations may be slightly increased.

6. Conclusion

A parallel implementation of the overlapping domain decomposition dual reciprocity method based on the block Jacobi method was presented. We observed that dividing the domain into p subdomains and employing p processors, overall, lead to p^2 times or even more efficiency over using a single domain with one processor. This was due to the fact that applying the DDM with p subdomains and using one processor, itself, improve the computational efficiency at least p times. Therefore, the contribution of the parallel implementation with p processors could be evaluated and it was about p times efficiency over using a single processor.

The DDM and the parallel implementation was accomplished for a rectangular domain. In more general cases, one needs to decide how to equally divide the domain and define the overlap. In such cases, we expect a similar improvement in the accuracy and computational efficiency. We leave this for a future work. Although extending the proposed method to the case of three dimension looks straightforward, it needs special consideration and we leave this for a future work. In this case, we expect more computational efficiency, since partitioning the domain, for example in the case of a cubic domain, is carried out in the three coordinate directions and this considerably increases the number of subdomains. Therefore, the contribution of the domain partitioning is increased (see Fig. 1b). However, the contribution of the parallel implementation will be affected due to the more communications between the processors.

References

[1] L.I. Cronsjo, Advances in Parallel Algorithms, Blackwell, Oxford, 1992.
[2] T.J. Dekker, W. Hoffimann, K. Prota, Parallel algorithms for solving large linear systems, J. Comput. Appl. Math. 50 (1994) 221–232.
[3] M.R. Dubal, Domain decomposition and local refinement for multiquadric approximations 1: second-order equation in one-dimension, J. Appl. Sci. Comput. 1 (1994) 146–171.
[4] K. Erhart, E. Divo, A.J. Kassab, A parallel domain decomposition boundary element method approach for the solution of large-scale transient heat conduction problems, J. Eng. Anal. Boundary Elem. 30 (2006) 553–563.
[5] W.F. Florez, H. Power, F. Chejne, Multi-domain dual reciprocity BEM approach for the Navier–Stokes system of equations, Commun. Numer. Meth. Eng. 16 (2000) 671–681.
[6] W.F. Florez, H. Power, F.C. Janna, Multi-domain dual reciprocity for solution of inelastic non-Newtonian flow problems at low Reynolds number, Comput. Mech. 27 (2001) 396–411.
[7] W.F. Florez, H. Power, DRM multi-domain mass conservative interpolation approach for the BEM solution of the two-dimensional Navier–Stokes equations, Comput. Math. Appl. 43 (2002) 457–472.
[8] M.A. Golberg, C.S. Chen, The theory of radial basis functions applied to the BEM for inhomogeneous partial differential equations, Boundary Elem. Commun. (1994).

[9] W. Gropp, E. Lusk, A. Skjellum, Using MPI, second ed., MIT Press, USA, 1999.
[10] E.J. Kansa, R.E. Carlson, F. Chejne, Radial basis functions: a class of grid-free, scattered data approximations, Comput. Fluid Dyn. J. 3/4 (1995) 479–496.
[11] S.R. Karur, P.A. Ramachandran, Radial basis function approximation in dual reciprocity method, Math. Comput. Model. 20 (7) (1994) 59–70.
[12] D. Nardini, C.A. Brebbia, A new approach to free vibration analysis using boundary elements, in: C.A. Brebbia (Ed.), Boundary Element Methods in Engineering, Spring-Verlag, Berlin, 1982.
[13] P.W. Partridge, C.A. Brebbia, L.C. Wrobel, The Dual Reciprocity Boundary Element Method, Computational Mechanics Publications, UK, 1992.
[14] V. Popov, H. Power, A domain decomposition in the dual reciprocity approach, Boundary Elem. Commun. 7 (1996) 1–5.
[15] V. Popov, H. Power, The DRM–MD integral equation method, Int. J. Numer. Meth. Eng. 44 (1999) 327–353.
[16] M.J.D. Powell, The theory of radial basis function approximation in 1990 in advances in numerical analysis II: wavelets, subdivision algorithms and radial basis functions, in: W. Light (Ed.), Oxford University Press, Oxford, UK, 1992, pp. 105–210.
[17] H. Power, R. Mingo, The DRM subdomain decomposition approach to solve the two-dimensional Navier–Stokes system of equations, Eng. Anal. Boundary Elem. 24 (2000) 107–119.
[18] M. Ramsak, L. Skerget, A multidomain boundary element method for two equation turbulence models, J. Eng. Anal. Boundary Elem. 29 (2005) 1086–1103.
[19] K. Shanazari, K. Chen, An overlapping domain decomposition for the dual reciprocity method, J. Eng. Anal. Boundary Elem. 27 (2003) 945–953.
[20] K. Shanazari, K. Chen, A parallel non-overlapping domain decomposition dual reciprocity method, in: Proceedings of Fifth Conference on Boundary Integral Methods, The University of Liverpool, UK, 2005, pp. 100–105.
[21] B.F. Smith, P.E. Bjorstad, W.D. Gropp, Domain Decomposition Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 1996.
[22] J. Verkaik, H.X. Lin, A class of novel parallel algorithms for the solution of tridiagonal systems, Parallel Comput. 31 (2005) 363–387.
[23] H. Wendland, Piecewise polynomial positive definite and compactly supported radial function of minimal degree, Adv. Comput. Math. 4 (1995) 389–396.