# Node Adaptive Domain Decomposition Method by Radial Basis Functions

**Pedro González-Casanova,**[1] **José Antonio Muñoz-Gómez,**[2]
**Gustavo Rodríguez-Gómez**[3]

[1] *Unidad de Investigación en Cómputo Aplicado, DGSCA, Universidad Nacional Autónoma de México, Insurgentes Sur No. 3000 Zona Cultural, Ciudad Universitaria C. P. 044510, D.F., México*

[2] *Universidad de Guadalajara, Centro Universitario de la Costa Sur, Av. Independencia Nacional 151, C.P. 48900, Autlán de Navarro, Jalisco, México*

[3] *Ciencias Computacionales, Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, Tonantzintla, México*

*Received 14 December 2007; accepted 1 September 2008*
*Published online 30 December 2008 in Wiley InterScience (www.interscience.wiley.com).*
*DOI 10.1002/num.20410*

During the last years, there has been increased interest in developing efficient radial basis function (RBF) algorithms to solve partial differential problems of great scale. In this article, we are interested in solving large PDEs problems, whose solution presents rapid variations. Our main objective is to introduce a RBF dynamical domain decomposition algorithm which simultaneously performs a node adaptive strategy. This algorithm is based on the RBFs unsymmetric collocation setting. Numerical experiments performed with the multiquadric kernel function, for two stationary problems in two dimensions are presented. © 2008 Wiley Periodicals, Inc. Numer Methods Partial Differential Eq 25: 1482–1501, 2009

*Keywords: domain decomposition; meshfree methods; node adaptive; partial differential equations; radial basis function*

## I. INTRODUCTION

During the last decade, radial basis function methods for the solution of partial differential equations have gained considerable attention both in the engineering and the mathematical communities. Several approaches for the solution of PDEs by RBFs have recently appeared in literature. Among them, we emphasize the pioneering works of Kansa [1, 2] on asymmetric collocation methods, the symmetric collocation version [3], the method of fundamental solutions [4], the Galerkin techniques for compact RBF [5], and the differential quadrature methods [6].

*Correspondence to:* Pedro González-Casanova, Unidad de Investigación en Cómputo Aplicado, DGSCA, Universidad Nacional Autónoma de México, Insurgentes Sur No. 3000, Zona Cultural, Ciudad Universitaria C. P. 044510, D.F., México (e-mail: pedrogc@dgsca2.unam.mx)

All these methods have already had an impact in different applied fields such as, financial mathematics, meteorology, geophysics, and computational fluid dynamics, see [7, 8] and the references therein. Unlike classical methods, such as finite differences, finite volumes, and finite elements, the lack of grid generation in these methods is a fundamental issue that explains their success. Among the most important advantages of these RBF methods we have that: they are are algorithmically simpler than the classical techniques, they are better suited to cope with problems in high spatial dimension with complex boundaries, the adapting techniques are far simpler and flexible than the corresponding mesh adaptive methods, and an exponential rate of convergence can be attained as the number of nodes increases [9].

On the other hand, it is well known that the condition number of RBFs collocation methods becomes highly ill conditioned when the number of nodes increases. This problem, known as the uncertainty principle of Schaback [10], has been tackled by many of different methods among which the most prominent are the preconditioning and the domain decomposition techniques.

In this article, we are interested in solving large PDE problems whose solution presents high gradients in some regions of the domain. Specifically, we introduce a RBF collocation algorithm that combines a domain decomposition technique with a node adaptive algorithm. The domain decomposition method is proposed to cope with the bad conditioning because of the large size of the PDE problem. Simultaneously, a node adaptive technique is built to efficiently approximate the oscillatory or high gradient regions of the solution. Several advances in domain decomposition and node adaptive methods have been reported in the literature. It is worth mentioning that in the field of approximation theory, Iske and Levesley [11] combines an adaptive node technique with a domain decomposition method for scattered data distributions.

In particular, domain decomposition methods for unsymmetric RBF collocation techniques have been treated in several recent works. Stationary PDE problems have been studied in [12–15], whereas time dependent problems have been treated in [16–18]. In a recent work by Zhou et al. [19], both multiplicative and additive RBF algorithms are formulated and numerically studied. Among their conclusions, the authors report that the multiplicative Schwarz algorithm is twice as fast as the additive one, although the additive version is easier to parallelize than the multiplicative version. In Zhang and Tan [20], a convergency proof is given for both RBFs multiplicative and additive Schwarz algorithms in two domains. Except for [16], which uses a multizone algorithm for nonuniform nodes, all these works have been formulated for Cartesian grids. In our case, a nonregular data distribution of nodes and a dynamic subdomain partition are necessary due to influence of the node adapting algorithm.

In the field of RBFs, node adaptive techniques some articles have also recently appeared in literature. In one dimension, both stationary and time dependent PDE problems have been treated.

In two dimensions, Behrens et al. [21, 22] formulate a RBF node adaptive technique. In this last approach, which is of semi-Lagrangian type, the authors use a local error estimate to determine the node insertion/remotion strategy of the algorithm. A Delaunay triangulation, and its dual, the Voronoi tessellation, is employed to localize a node and its neighboring points. In Drisscoll and Heryudono [23], the authors formulated a node adaptive technique based on a global error estimate that determines the node insertion/remotion strategy. In Pereyra et al. [24], a global optimization technique is formulated to solve stationary PDEs problems by the collocation technique based on Gaussian kernels.

As far as the authors know, none of the former methods combine both domain decomposition and node adapting techniques to solve the PDE problems. The method presented in this article is based on a RBF Schwarz multiplicative domain decomposition algorithm with a variable number of subdomains combined with a local node adaptive technique based on a quadtree structure.

TABLE I. Classical globally supported radial basis functions.

| | |
|---|---|
| Multiquadric (MQ) | $\phi(r) = \sqrt{r^2 + c^2}$ |
| Inverse multiquadric (IMQ) | $\phi(r) = 1/\sqrt{r^2 + c^2}$ |
| Gaussian (GA) | $\phi(r) = e^{-(cr^2)}$ |
| Thin-plate splines (TPS) | $\phi(r) = r^m \log r, m = 2, 4, 6, \dots$ |
| Smooth splines (SS) | $\phi(r) = r^m, \quad m = 1, 3, 4, \dots$ |

This article is organized as follows: In Section II, a brief review of the asymmetric collocation method is presented. In Section III, we formulate both overlapping additive and multiplicative Schwarz algorithms in the context of RBFs. Section IV, is devoted to introduce a node adaptive algorithm for non-uniform centers. In Section V, the domain decomposition node adaptive algorithm is presented. Numerical results for a Poisson and a stationary convection-diffusion problems are presented in Section VI. In Section VII, final remarks and conclusions are presented.

## II. RADIAL BASIS FUNCTIONS

Consider the following generic elliptic PDE problem

$$\mathcal{L}u = f \quad \text{in} \quad \Omega \subset \mathbb{R}^d, \quad \mathcal{B}u = g \quad \text{on} \quad \partial\Omega, \tag{2.1}$$

where $\mathcal{L}$ is a linear partial differential operator and $\mathcal{B}$ a boundary operator that can be of Dirichlet, Neumann, or Robin type. In Kansa's approach [1,2], the exact solution $u(x)$ of problem (2.1) is approximated by ansatz

$$\tilde{u}(x) = \sum_{j=1}^{N} \lambda_j \phi(\|x - x_j\|) + p(x), \quad x \in \mathbb{R}^d, \tag{2.2}$$

where $\lambda_j$ is the unknown coefficients and $p \in \pi_m^d$ is a polynomial, which depends on the particular RBF kernel [7], of degree at most $m$ in $d$-dimensions. The most popular RBFs are displayed in Table I, where $c$ is a positive constant called the shape parameter. Substituting the ansatz (2.2) in the elliptic PDE problem (2.1), where we take $p = 0$ for exposition simplicity, we obtain the following algebraic linear system of equations

$$A\Lambda = F, \tag{2.3}$$

where $F = [f \quad g]^T$, $A = [\mathcal{L}\Phi_a \quad \mathcal{B}\Phi_b]^T$, $\Lambda \in \mathbb{R}^N$, $\Phi_a \in \mathbb{R}^{n_i \times N}$, $\Phi_b \in \mathbb{R}^{n_f \times N}$, and $N = n_i + n_f$ denotes the division of the $N$ collocation nodes into interior and boundary nodes. The entries of the matrix $A$ are $\mathcal{L}\Phi_a = \{\mathcal{L}\phi(\|x_i - x_j\|)\}$, $i = 1, \dots, n_i$ and $\mathcal{B}\Phi_b = \{\mathcal{B}\phi(\|x_i - x_j\|)\}$, $i = n_i + 1, \dots, N$, $j = 1, \dots, N$. In the literature, this method is also known as the unsymmetric collocation method, due to the asymmetry of the matrix $A$.

The system (2.3) can be solved by $LU$ factorization to obtain the vector $\Lambda = (\lambda_1, \dots, \lambda_N)^T$, which determines the numerical solution (2.2) at any point in $\Omega$. Namely, we obtain that $\tilde{u} = H\Lambda$, where each row of $H \in \mathbb{R}^{M \times N}$ is given by $(\phi_1, \dots, \phi_N)$ with $\phi_j = \phi(\|x - x_j\|)$ and where $x$ belongs to the set $\{x_i\}_{i=1}^M$ of points where the solution is evaluated. To the best of the authors knowledge, no theoretical proof exists of the invertibility of the matrix $A$. However, the method has been successfully applied to a wide range of applications.

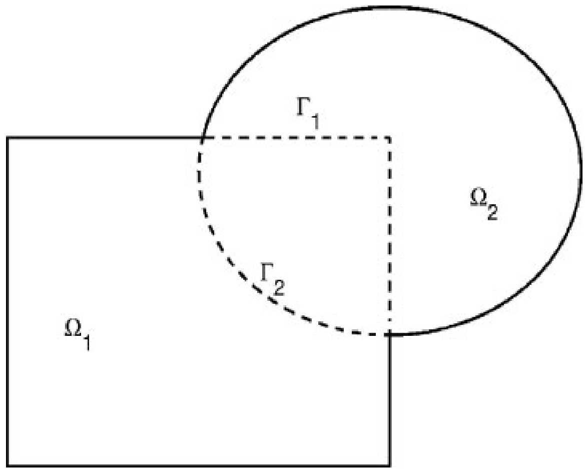

FIG. 1. Two subdomains with overlapping.

## III. OVERLAPPING DOMAIN DECOMPOSITION METHOD

Domain decomposition methods are divide-and-conquer techniques that are used to solve large PDE problems. The original problem is divided into small subproblems whose solutions are joined through an iterative process to build an approximate global solution. In this section, we present Schwarz overlapping methods in the context of RBFs collocation techniques.

To do so, recall first that the classical Schwarz alternating method is an iterative algorithm that solves the subproblem in each subdomain alternately and couples the subproblems together by overlapping each subdomain and exchanging function values (Dirichlet data) at the artificial boundaries.

Although the following approach can be easily extended to $N$ subdomains, for simplicity on the exposition, we define the alternating Schwarz overlapping method for the case where the whole domain $\Omega$ is decomposed into two subdomains $\Omega_1$ and $\Omega_2$ (i.e., $\Omega = \Omega_1 \cup \Omega_2$ with the two artificial boundaries $\Gamma_1$, $\Gamma_2$ intersecting $\partial\Omega$, see Fig. 1). To illustrate the method, consider an elliptic PDE with Dirichlet boundary data

$$\begin{aligned} \mathcal{L}u &= f \quad \text{in} \quad \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{3.1}$$

where $\mathcal{L}$ is some elliptic differential operator. In what follows, we summarize the Schwarz additive and multiplicative domain decomposition techniques to formulate these schemes within the RBF setting. In the *additive version*, the artificial boundary values of each subdomain problem at the $(n + 1)$th step are updated from the results of the $n$th step, whereas in the *multiplicative version*, the artificial boundary values of each subdomain are updated from the most recent results obtained from its neighboring subdomains. So, we could properly say that the additive method is a Jacobi type iteration, whereas the multiplicative one is a Gauss-Seidel.

- *Additive version*: Set $u_i^0$ initially in $\Omega_i$, $i = 1, 2$, and construct $u_i^n$ in parallel:

$$\begin{aligned}
\mathcal{L}u_1^{n+1} &= f \quad \text{in } \Omega_1 \\
u_1^{n+1} &= g \quad \text{on } \partial\Omega_1 \backslash \Gamma_1 \\
u_1^{n+1} &= u_2^n \quad \text{on } \Gamma_1 \\
&\text{and} \\
\mathcal{L}u_2^{n+1} &= f \quad \text{in } \Omega_2 \\
u_2^{n+1} &= g \quad \text{on } \partial\Omega_2 \backslash \Gamma_2. \\
u_2^{n+1} &= u_1^n \quad \text{on } \Gamma_2
\end{aligned} \tag{3.2}$$

$n = 0, 1, 2, \ldots$

- *Multiplicative version*: Set $u_1^0$ initially in $\Omega_1$ and construct $u_1^{k+1/2}$ in $\Omega_1$ and $u_2^{k+1}$ in $\Omega_2$ sequentially:

$$\begin{aligned}
\mathcal{L}u_1^{k+1/2} &= f \quad \text{in } \Omega_1 \\
u_1^{k+1/2} &= g \quad \text{on } \partial\Omega_1 \backslash \Gamma_1 \\
u_1^{k+1/2} &= u_2^k \quad \text{on } \Gamma_1 \\
&\text{and} \\
\mathcal{L}u_2^{k+1} &= f \quad \text{in } \Omega_2 \\
u_2^{k+1} &= g \quad \text{on } \partial\Omega_2 \backslash \Gamma_2. \\
u_2^{k+1} &= u_1^{k+1/2} \quad \text{on } \Gamma_2
\end{aligned} \tag{3.3}$$

$k = 0, 1, 2, \ldots$

The solution $u^n$ in $\Omega$ can be composed in many ways from $u_1^n$ and $u_2^n$ such that $u^n$ is smooth enough and $u^n = u_i^n$ in $\Omega_i \backslash (\Omega_1 \cap \Omega_2)$.

Using the notation introduced in Section II, choosing the subdomain problems small enough, and assuming the invertibility of $A_i$, we have that their solutions can be directly obtained from

$$\Lambda_i = A_i^{-1} F_i, \quad i = 1, 2, \tag{3.4}$$

where $A_i = [A_{\bar{\Omega}_i \backslash \Gamma_i} \; A_{\Gamma_i}]^T$, $F_i = [F_{\bar{\Omega}_i \backslash \Gamma_i} \; F_{\Gamma_i}]^T$, and $\bar{\Omega}_i = \Omega_i \cup \partial\Omega_i \cup \Gamma_i$ denote the closure of the domain. The solution $\tilde{u}_i$ is given by

$$\tilde{u}_i = H_i \Lambda_i, \quad i = 1, 2, \tag{3.5}$$

where $H_i$ is the matrix of RBFs at collocation points where the solution is requested.

Note that $F_{\Gamma_i} \tilde{u}_i := I_{\Omega_i \to \Gamma_j} \tilde{u}_i$, where $I_{\Omega_i} \to \Gamma_j$ is the discrete operator that assigns the nodes of $\Omega_i$ to the nodes on the curve $\Gamma_j$. Clearly, $F_i$ includes the artificial boundary conditions that changes with the updating results of its neighboring subdomains. At each $(n+1)$th iterative step, the subdomain solution to (3.1) on $\bar{\Omega}_i$ can be rewritten as

$$\Lambda_i^{n+1} = \Lambda_i^n + A_i^{-1}\big(F_i - A_i \Lambda_i^n\big), \quad n = 0, 1, 2, \ldots. \tag{3.6}$$

Or using Eq. (3.5), we can restate Eq. (3.6) in terms of the approximate solution

$$\tilde{u}_i^{n+1} = \tilde{u}_i^n + H_i A_i^{-1}\left(F_i - A_i H_i^{-1}\tilde{u}_i^n\right), \quad n = 0, 1, 2, \ldots. \tag{3.7}$$

The Schwarz additive and multiplicative overlapping domain decomposition methods can now be formulated in terms of the radial basis function collocation setting as follows.

- *RBF additive version*: In this algorithm, the artificial boundary values of each subdomain problem at the $(n+1)$th step are updated from the results of the $n$th step. Hence, (3.7) can be written as

$$\tilde{u}_i^{n+1} = \tilde{u}_i^n + H_i A_i^{-1}\left(F_i^n - A_i H_i^{-1}\tilde{u}_i^n\right), \quad n = 0, 1, 2, \ldots. \tag{3.8}$$

  for $i = 1, 2$.
- *RBF multiplicative version*: In this algorithm, the artificial boundary value of each subdomain is updated from the most recent results obtained from its neighboring subdomains

$$\begin{aligned} \tilde{u}_1^{n+1/2} &= \tilde{u}_i^n + H_1 A_1^{-1}\left(F_1^n - A_1 H_1^{-1}\tilde{u}_1^n\right) \\ \tilde{u}_2^{n+1} &= \tilde{u}_2^{n+1/2} + H_2 A_2^{-1}\left(F_2^{n+1/2} - A_2 H_2^{-1}\tilde{u}_2^{n+1/2}\right), \quad n = 0, 1, 2, \ldots. \end{aligned} \tag{3.9}$$

It is worth noting that the additive (3.8) and multiplicative (3.9) Schwarz versions correspond to the block Jacobi and block Gauss-Seidel iterative type methods, respectively.

The former schemes allow us to calculate the approximate solution in each of the two subdomains $\Omega_1$ and $\Omega_2$. As mentioned, there are several ways to smoothly compose the global solution. In the case of Cartesian grids, the existence of a mesh parameter combined with a particular criteria in the overlapping zones gives us a solution to this problem. In this work, however, we aim to formulate a domain decomposition algorithm for scattered data centers combined with a knot adaptive strategy. Thus, neither the subdomain partition strategy nor the extension algorithm used to build the overlapping zones are trivial. The nonuniform data distribution in each subdomain requires a special criteria to form the extended subdomains. In the following section, we shall briefly describe the node adaptive strategy, whereas in Section V, we formulate the combined domain decomposition node adaptive algorithm.

## IV. ADAPTIVE THINNING METHOD

In this section, we construct an algorithm which combines Behrens et al. [21, 22] refinement method with a quadtree type algorithm in two dimensions. The main difference with respect to [21,22] is in the data structure used to localize and insert or remove the nodes. In this article, instead of using a Delaunay triangulation or its dual Voronoi tessellation to search and insert/remove the nodes, we use a quadtree type algorithm to perform these tasks. The main difference is that this type of data structure allows us, by means of an octree algorithm, to solve problems in 3D.

We first describe the error indicator function used to mark the nodes that will be refined. Then, we introduce the data structure used to perform the node refinement scheme for nonequally spaced data.

Let $\Xi$ denote the current node set that comes from the initial knot distribution. For each node $x \in \Xi$, we select a set of neighboring nodes $N_x \backslash x \in \Xi$. We construct a local interpolant $I\tilde{u}(x)$

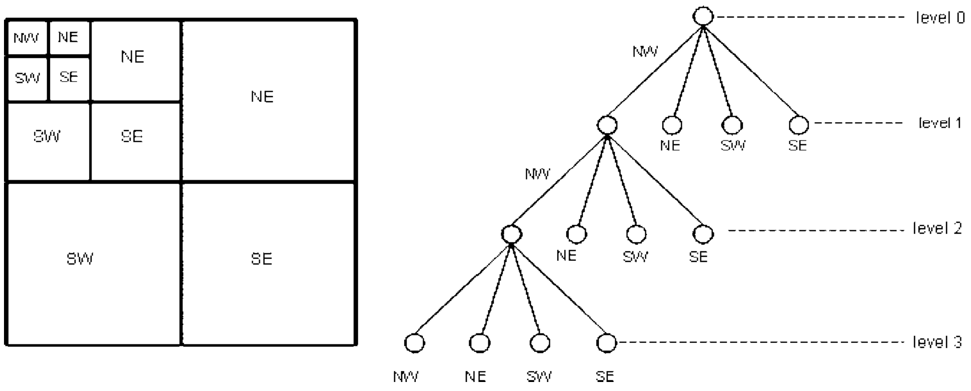

FIG. 2. Block and quadtree representation.

based on $N_x \backslash x$, using Eq. (2.2) plus a polynomial of first degree with a TPS $(r^2 \log r)$ as RBF kernel.

Based on the numerical solution $\tilde{u}(x)$ of the partial differential problem and the local interpolant $I\tilde{u}(x)$, we define the error indicator

$$\eta(x) = |\tilde{u}(x) - I\tilde{u}(x)|, \tag{4.1}$$

which assigns a significance value $\eta(x)$ to each node $x \in \Xi$. As local TPS interpolants reproduces linear polynomials, the value of $\eta(x)$ is small when the surrounding values of $\tilde{u}(x)$ belong to a smooth region of the solution. On the other hand, a high value $\eta(x)$ indicates that the numerical approximation around of $\tilde{u}(x)$ presents a sharp variation, which corresponds to regions where the PDE solution has a high gradient.

Based on the error indicator $\eta(x)$, we can now define the rule to flag the nodes to be refined or to be coarsened according to the following criteria.

**Definition.** *Let $\theta_r$ and $\theta_c$ be two threshold satisfying $0 < \theta_c < \theta_r$. We say that a node $x \in \Xi$ is flagged to be refined if $\eta(x) > \theta_r$, alternatively if $\eta(x) < \theta_c$ the node is marked to be removed.*

This definition can be understood in the following way: only the nodes that have an error larger than a threshold value are marked to be refined. In a similar way, the nodes that have an error lower than a threshold value are marked to be removed. We note that according to our experience, the threshold values can be chosen in such a way that $\theta_c$ is approximately of two orders of magnitude less than the selected value of $\theta_r$. In general, however, this is a problem dependent election, which up to now do not have an abstract support, see [21,22].

The quadtree is a hierarchical spatial data structure [25,26] that applies when the domain $\Omega$ is a square or a rectangle. The idea is to decompose the domain recursively into four new elements; these new elements are named children of the father element. Each children is labelled with regards to its relative position as {NW, NE, SW, SE}. The center of each element or quadrant contains a node. Figure 2 shows the box and quadtree representation corresponding to three levels of refinement.

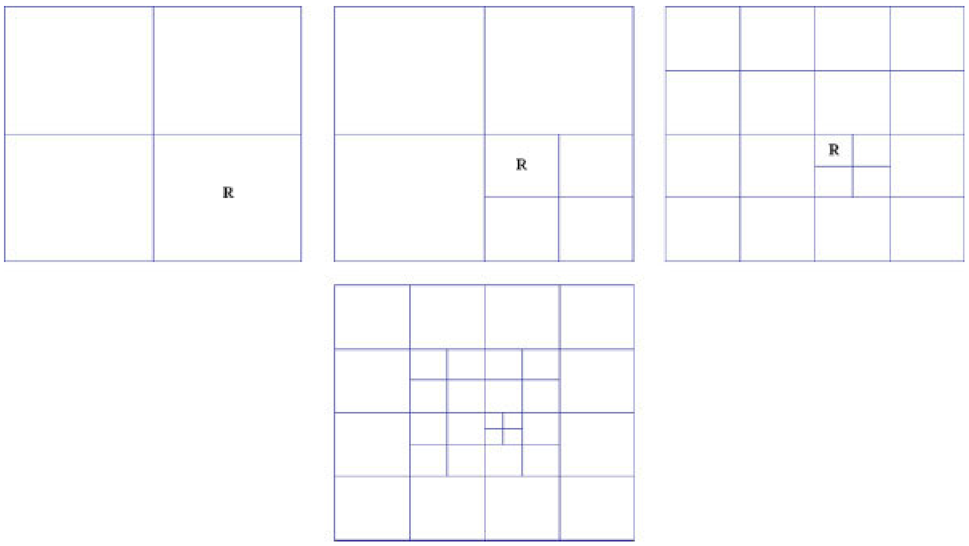

FIG. 3. Illustration of node refinement in two dimensions, it is observed that the neighboring cells are refined by the constrain 2:1. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

The information of the quadtree is stored in a tree structure, which contains the node $x \in \mathbb{R}^2$, the numerical solution $\tilde{u}(x)$, the shape parameter $c$, the refinement level, the size of the box, the type of the node {NW, NE, SW, SE}, a pointer to its father, and a pointer to its four children. Based on the last three items, the neighbors

$$N_x = \{\mathrm{N}, \mathrm{S}, \mathrm{W}, \mathrm{E}, \mathrm{NW}, \mathrm{NE}, \mathrm{SW}, \mathrm{SE}\}, \tag{4.2}$$

of a node can be determined in an efficient way [25]. Note, that the local thin-plate spline interpolant, necessary to define the error estimator $\eta(x)$, is based on the set of points given by (4.2).

After each refinement, the leave of the tree must satisfy the 2:1 relationship at the bottom of the tree. That is, for each node, we insert new nodes to have a similar ($\pm 1$) refinement level on its neighbors. This 2:1 restriction is required to diminish the highly numerical oscillations near the regions where the solution is rapidly varying.

Based on the local error indicator $\eta(x)$, we flag the elements to be refined. The refinement process is illustrated in Fig. 3 for an initial quadtree with one level of refinement corresponding to four elements. For each element marked for refinement (denoted by R in the figure), four elements are inserted. Observe in Fig. 3 that the (unmarked) neighborhoods of the elements marked to be refined are also refined. This corresponds to the imposed 2:1 constrains in the level of refinement. A similar procedure is used to remove a given node. This strategy allows us to control the knot density distribution, in such a way that it smoothly decreases from high density regions to zones where the knot density is low. This procedure allows us avoid the possible presence of Gibbs phenomena that can be generated at a sharp discontinuity between two different nodes density regions.

## V. LOCAL NODE REFINEMENT AND DOMAIN DECOMPOSITION ALGORITHM

In Section III, the basic Schwarz algorithms for RBF in two domains were formulated, whereas in Section IV, a node insertion/remotion technique for scattered centers was presented. Here, we combine both techniques to build a domain decomposition node refinement algorithm for nonuniform data. We first explain how to build the partition of the domain in such a way that it can dynamically change as the node distribution is modified. Then, we explain how to build the extended subdomains for this dynamic partition of nonuniform nodes. A load balance formulation will be also presented. We explain how to use the multiplicative Schwarz algorithm within this setting. Finally, we explain how the node adaptive technique explained in Section IV is used.

We start by pointing out that the collocation scheme to be used is based in multiquadric kernels. Thus, after a first node distribution of the domain $\Omega$—which might or not be Cartesian—is built, an initial shape parameter $c_{\text{ini}}$ must be given. Although in this work, all the numerical examples are based on a first or initial coarse Cartesian grid, the algorithm can be initialized by using a first nonuniform sparse and evenly distributed set of nodes. Within this work, we have used a constant value for the initial shape parameter $c_{\text{ini}}$.

The first step of the domain decomposition node adaptive technique, Algorithm 1, is to built a quadtree index structure that records a unique global index related to each node of the domain.

The second step is to use a coordinate bisection algorithm based on the former quadtree structure, see [27, 28], to divide the domain. Recall that this technique is a divide and conquer procedure. For completeness, we summarize the basic steps of this algorithm. First, the coordinate direction of longest expansion of the domain is determined, which without loss of generality, we can assume to be the $x$ direction. Then all vertices are sorted according to their $x$ coordinate. Half of the vertices with small $x$-coordinates are assigned to one domain, the other half with the large $x$-coordinates are assigned to the second domain. For each new subdomain, the ordering process is repeated but now in the $y$ direction. This divide and conquer procedure is repeated recursively. As a QuickSort algorithm is used, the numerical complexity is of order $O(N \log N)$, where $N$ is the number of nodes. Note that each subdomain $\Omega_i$, $i = 1, \dots, P$ forms a cover of disjoint sets, $\Omega_i \cap \Omega_j = \emptyset$ if $i \neq j$, of $\Omega$. To keep the load balance of the subdomains, before the division procedure is performed, the algorithm verifies that each subdomain has at least a minimum number of $N_{\max}$ nodes. Only if this condition is fulfilled a new subdivision is performed. This condition guaranties that there is no domain which has more than $N_{\max}$ nodes. This is how the dynamical subdomain adaptation is done. Note that the maximum number of nodes $N_{\max}$ depends on the condition number of the collocation matrix related to each subdomain. As we are not using preconditioning techniques, typical values of $N_{\max}$ which give reasonable condition numbers are within the range of $10^3$.

Once the subdomain partition has been performed and to apply the Schwarz multiplicative overlapping algorithm, an extended partition has to be built to form the overlapping regions and the artificial boundaries. For Cartesian grids, this is a simple task that is very similar to the one used by the finite difference method. However, for nonuniform data distributions—which is our case—the criteria becomes more involved. In this case, we aim that the nodes of the overlapping regions that correspond to a sharp gradient part of the solution have a higher level of expansion than those which are placed within a smooth region. This is the criteria that we shall use to build the overlapping regions for the nonuniform node case.

To construct these overlapping regions, we first need to perform a local copy of the global indexes which corresponds to the nodes related to each subdomain. Thus, we build a set of lists containing each the set of local indexes related to the nodes of each subdomain.

TABLE II. Levels of expansion to built the overlapping zones.

| Refinement level | Expansion level |
|---|---|
| 5–6 | 2 |
| 6–10 | 3 |
| 11–14 | 4 |

Recall now that the level of the tree structure is determined by the node refinement algorithm. The deeper the level of a given node in the tree, the higher the refinement level. Moreover, as the knot insertion is determined by the thinning algorithm, deeper nodes in the tree correspond to regions of the solution that presents sharper gradients. Thus, the knot tree structure determines whether a knot belongs to a smooth region or to a sharp gradient part of the solution. These observations are behind the criteria that we shall use to expand the subdomains.

The overlapping regions are built by means of an iterative procedure that we divide into two steps: extension and expansion of the subdomains. The extension step depends on the nearest neighboring nodes, related to a subdomain $\Omega_i$, which are used to build a first set of overlapping regions for this domain. These regions do not depend on the gradient of the solution, a criteria which as mentioned, is at the core of the algorithm. The expansion step solve this problem by using the tree structure information, that is related to the gradient of the solution. In this step, the algorithm determine—by using the position in the tree—the number of nodes that should be expanded for each node that belongs to the overlapping regions built in the extension step.

A specific criteria should be used for this purpose. By means of extensive numerical experimentation, we found that the criteria formulated in Table II gave excellent results.

We now explain in detail how to formulate these two steps. In the first step, given a subdomain $\Omega_i$, we build the overlapping region by asking if for a given fixed node its nearest neighboring nodes $N_x$, see (4.2), belong to $\Omega_i$. If these nodes do not belong to this subdomain, then $\Omega_i$ is extended by including these new nodes in the subdomain:

$$\text{if } (N_x(\Omega_i) \notin \Omega_i) \text{ then } \Omega_i^e \longleftarrow N_x(\Omega_i),$$

$\Omega_i^e$ conform the overlapping region and the new subdomain is now formed by: $\Omega_i = \Omega_i^e \cup \Omega_i$. Recall that $\Omega_i^e$ could contain repeated nodes, so before inserting a new node in $\Omega_i^e$ we determine, by a binary search over its local index list, whether the new node does belong or not to $\Omega_i^e$, if it does not belong we include this node.

In the second step, based on the level of the tree corresponding to the set of nodes that belong to $\Omega_i^e$, we use the criteria given in the second column of Table II, to iteratively expand each node of $\Omega_i^e$. The artificial boundary $\Gamma_i$ of the extended domain is formed by the last two expanded nodes.

The Schwarz multiplicative algorithm given by (3.9), which can be easily extended to P subdomains, can now be applied to the extended partition. Namely, the multiplicative Schwarz algorithm given by (3.9), which can be easily extended to $P$ subdomains is used. Although the number of subdomains P changes in each global iteration, see Algorithm 1, it should be observed that the multiplicative algorithm can be directly applied within each inner loop. The stopping criteria for the Schwarz algorithm is given by: $\|\tilde{u}^k - \tilde{u}^{k-1}\| \leq 10^{-3}$, calculated on the artificial boundaries. As it is well known, when the range of the solution lies out of the interval (0,1), it is recommendable to use the relative error instead of the maximum error. Note that before applying the Schwarz algorithm, it is necessary to localize the interior nodes of each subdomain that will be used to update the artificial boundary of the neighboring subdomains. This is efficiently performed by means of the local indexes lists.

The integration of the subdomain solutions to form the global solution, corresponding to the domain $\Omega$, is performed by taking the average values of the possible intersecting points in the overlapping regions.

Once the global solution has been built, it is then refined through the thinning algorithm presented in Section IV. The value of the shape parameter $c$ is related to its corresponding level of the quadtree. The deeper in the tree the lower value of $c$. Thus, we determine that a useful criteria to obtain the new value of $c$ for each refinement node is $c_{\text{new}} = c_{\text{old}}/2$. This technique is thus an $h - c$ type algorithm.

An initial level $level_{\text{ini}}$ determines whether a given node will be removed or not: a node can be removed only if its level in the tree is greater that $level_{\text{ini}}$. This condition is introduced to avoid the formation of regions without nodes. In this context, reasonable values of $level_{\text{ini}}$ are four or five, which means that the Cartesian initial grid or nonuniform evenly data distribution, have $4^4$ or $4^5$ nodes, respectively. Note also that, the deeper the level in the tree, the closest the distances between the nodes. Consequently, to control the minimum distances between the nodes it is necessary to bound the maximum number of levels in the refinement procedure. In our experience, this can be achieved by limiting the number of levels to be 9 or 11.

The global stopping criteria of the algorithm in the global loop is given by the following rule: either there are no more nodes to insert or a maximum number of iterations $N_{\text{iter}}$ is exceeded.

**Algorithm 1**. Local node refinement and domain decomposition algorithm

```
Define the variables: $level_{\text{ini}}, c_{\text{ini}}, \theta_r, \theta_c, N_{\text{max}}$.
Build the initial tree and refine twice on the boundary
while $\max\{\eta(x)\} > \theta_r$ or iteration$< N_{\text{iter}}$ do
    1. Partition $\Omega$ obtaining the subdomains $\Omega_i, i = 1, \dots, P$.
    2. Form the overlapping zones.
    3. Solve the PDE problem by the multiplicative Schwarz algorithm.
    4. Join the solution obtained in each $\Omega_i$ to build the solution in $\Omega$.
    5. Perform the node adaptive scheme.
end while
```

## VI. NUMERICAL RESULTS

The aim of this section is to demonstrate the algorithms described earlier for the solution of PDEs. For this purpose, we consider two partial differential equations in two dimensions. The first is a Poisson equation and the second is a convection-diffusion problem. Although the numerical results presented in this section have been performed by using the DD-refinement algorithm with multiquadric kernels, we note that different algebraic RBF functions have been successfully used, for a DD algorithm using an equispaced grid, to solve PDE problems, see [13]. For this reason, we have included numerical experiments that use this kind of kernels within the DD-refinemet algorithm, to compare them with the multiquadric results.

### A. Poisson

Consider the following Poisson problem in two-dimensions

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f, & \text{in } \Omega, \\ \qquad u \qquad = g, & \text{on } \partial\Omega, \end{cases} \tag{6.1}$$

TABLE III. Decreasing of the error vs. iterations for Poisson problem.

| Iteration | Subdomains | $N$ | $E_{\max}$ |
|---|---|---|---|
| 0 | 2 | 2152 | 13.38180 |
| 1 | 2 | 2884 | 3.078997 |
| 2 | 4 | 3589 | 0.108308 |
| 3 | 4 | 5344 | 0.010537 |
| 4 | 8 | 7252 | 0.010239 |
| 5 | 8 | 8548 | 0.006888 |
| 6 | 8 | 8686 | 0.006892 |
| 7 | 8 | 8728 | 0.006890 |

where $u = u(x, y)$, $\Omega$=$[-1, 1]^2$ and the functions $f, g$: $\mathbb{R}^2 \to \mathbb{R}$ are determined using the analytical solution

$$u(x, y) = \tanh(x + 50y). \tag{6.2}$$

The analytical solution presents a sharp variation parallel to the $x$-axis with $y = 0$, whereas outside this zone it takes a constant value. Our goal is to reduce the numerical error and simultaneously increase the throughput by using Algorithm 1. For this purpose, the following parameters where selected: initial shape parameter $c_{\text{ini}} = 0.1$, coarse node distribution $N = 4^5$ plus two additionally levels of refinement on the boundary. A maximum of 1500 nodes in each subdomain without overlapping region is allowed, and threshold values relative to the coarse and node refinement are $\theta_c = 0.0005$ and $\theta_r = 0.01$, respectively.

In Table III, we show how the error in the numerical approximation is reduced as Algorithm 1 is iterated; see Section V. In the first column, the iteration number is shown, the subdomain adaptivity is depicted in the second column, the number of nodes at each iteration is displayed in the third column. The fourth column corresponds to the measurements of the error using the norm $\|\cdot\|_\infty$ calculated over the whole domain $\Omega$.

It is observed from the fourth column of Table III that, as the number of iterations increases the error diminishes, thus reaching the objective of reducing the approximation error by means of an adaptive local node refinement. On the other hand, note that after the fourth iteration the error does not present a considerable reduction. This is consistent with the well-known fact that for FD and FE methods the rate of convergence is reduced as the number of subdomains is increased, unless a proper two-level Schwarz method is used. Although we are not using a two-level Schwarz technique in this work, we stress that the inclusion of an adaptive technique—which is the purpose of this work—within the domain decomposition method implies that the throughput of the scheme is increased. This fact will be explicitly illustrated in the next section.

This algorithm corresponds to an $h$-$c$ refinement scheme. The initial and final condition number were $10^7$ and $[10^8, 10^9]$, respectively. As the number of nodes increases, see third column, the number of subdomains also increases as it is shown in the second column. This allows us to handle large amounts of data by using the domain decomposition method and subdomain adaptivity in a computer cluster environment.

Figure 4 shows the numerical solution $\tilde{u}(x, y)$ at different iterations = $\{0, 1, 7\}$. It can be observed that the nodes were refined in an incremental way. The refined zones are near the axes $y = 0$ and parallel to the $x$-axis, which correspond to the zones where the greater spatial variation of the analytic solution exists.

As mentioned at the beginning of this section, we also solve the Poisson problem by using algebraic RBF. Specifically, we obtained that with the RFB $r^4 \log r$, the DD-adaptive Algorithm 1,

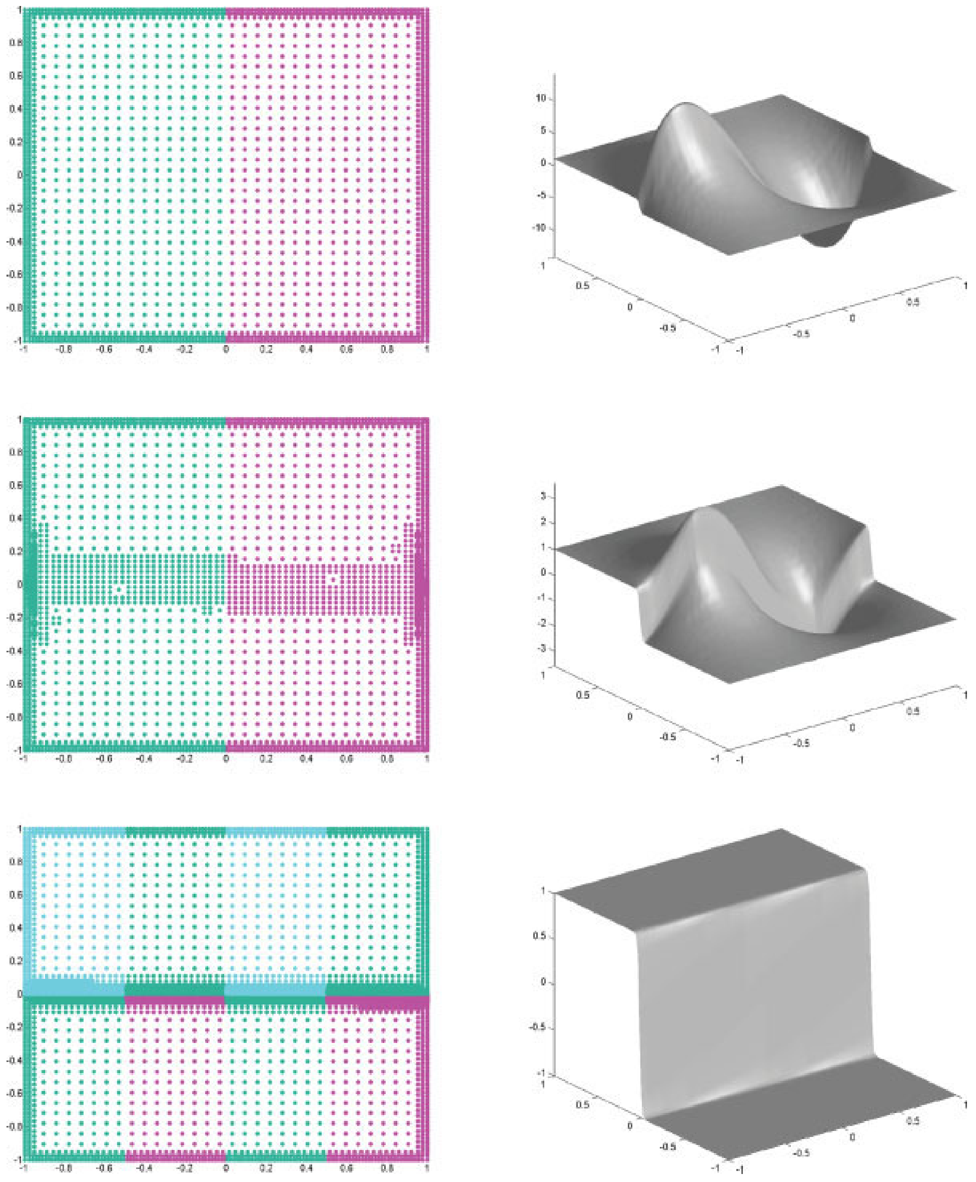

FIG. 4. Domain decomposition and adaptive node refinement for Poisson problem at iterations = {0,1,7}. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

gives an $E_{\max}$ error of 0.026567, using 8800 nodes and 7 iterations which is an order of magnitude poorer than the equivalent result obtained by the multiquadric function. On the other hand, we obtained that, the RBF: $r^5$ produced an $E_{\max}$ error of 0.007232, using 8752 nodes and 7 iterations, which is similar to the multiquadric error obtained by the same method. All these results, including the multiquadric results, were obtained by using the same values for the parameters used in Algorithm 1. We note that the equivalent results that we obtained by using the $r^7$ RBF kernel

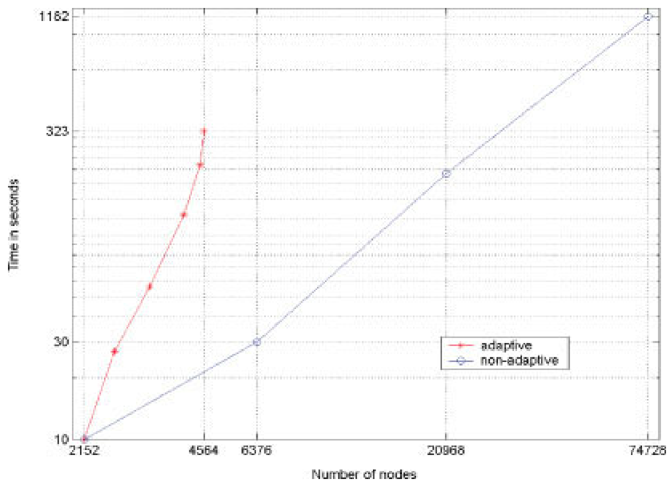

FIG. 5. Comparison between node adaptive and nonadaptive schemes. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

produced a blow up in the numerical solution. The explanation to this problem is that, due to the node adaptive procedure, the condition number of each subdomain is highly increased for higher degree radial basis functions. Although this also happen if we use an equispaced or Cartesian grid without node refinement, the condition number does not increase in such a dramatical way, thus giving reasonable results for the $r^7$ kernel.

***Adaptive vs. Nonadaptive Methods.*** In this experiment, we compare node adaptive vs. nonadaptive schemes by fixing the error in the $\| \cdot \|_\infty$ norm that both methods should reach. In both cases we use the multiplicative Schwarz method.

The program was written in C, and compiled with gcc version 3.3.4 with the flag -O2. Computations were carried out in a dedicated Intel Xeon (3.2 GHz, 4Gbytes RAM, under Linux). The time reported includes all the steps in the inner loop of Algorithm 1.

The test was carried out by using the free shape parameter thin-plate spline kernel function, $r^4 \log r$, plus a bivariate polynomial of total degree at most two. The following parameters were selected: a maximum error of $4 \times 10^{-2}$ was fixed to perform both experiments; the threshold values of the node refinement and coarse refinement were $\theta_r = 0.028$ and $\theta_c = 0.00001$, respectively.

Figure 5 shows two graphs in a loglog scale. In the $x-$axis the number of nodes are displayed, whereas the $y-$axis corresponds to the time expressed in seconds. The nonadaptive scheme relies on successive refinements of the leaves in the quadtree structure. Both methods start with the same initial node distribution. As we can see from Fig. 5, to meet approximately the same error, the node adaptive scheme combined with the domain decomposition method takes only 27% of the time required by the nonadaptive domain decomposition scheme. In addition, the final number of nodes for the node adaptive method were 4564, which represents an efficiency of 94% relative to the number of nodes used by the nonadaptive scheme.

It is well known in the field of FEM, FDM, and recently in the context of RBF [19], that the increment in the number of subdomains $P$ produces a reduction of accuracy and a decrease of the convergence rate, however, as pointed in the former section, the throughput of the scheme is increased.

TABLE IV. Decreasing of the error vs. iterations for convection-diffusion problem.

| Iteration | Subdomains | $N$ | $E_{\max}$ |
|---|---|---|---|
| 0 | 2 | 2152 | 0.364755 |
| 1 | 5 | 4540 | 0.082683 |
| 2 | 5 | 4351 | 0.091548 |
| 3 | 5 | 5077 | 0.081013 |
| 4 | 7 | 7285 | 0.084177 |
| 5 | 8 | 9472 | 0.080848 |
| 6 | 9 | 10,615 | 0.075643 |
| 7 | 9 | 10,969 | 0.072041 |
| 8 | 10 | 11,023 | 0.053759 |
| 9 | 10 | 11,029 | 0.053766 |

## B. Convection-Diffusion

Our goal in this section is to investigate if the proposed Algorithm 1 can dynamically adapt the nodes and the subdomains for a linear convection-diffusion problem. For this purpose, consider the following equation in two dimensions

$$\begin{cases} \beta\nabla^2 u + \mathrm{v}\cdot\nabla u = 0, & \text{in } \Omega, \\ u = g, & \text{on } \partial\Omega, \end{cases} \tag{6.3}$$

where $\Omega = [0, 0.5]^2$, $\nabla$ is the gradient differential operator, $\beta$ the diffusion coefficient, and $\mathrm{v} = [v_x, v_y]^T$ the advection coefficient (or the velocity) vector. The problem is solved by defining the analytical solution as

$$u(x, y) = a(e^{-c_x x} + e^{-c_y y}), \tag{6.4}$$

where $c_x = v_x/\beta$ and $c_y = v_y/\beta$. The following values $a = 0.5$, $v_x = v_y$ were used and the Péclet number is defined as $Pe = v/\beta$.

We fix the diffusion coefficient $\beta = 1$ and considered the case where the velocity coefficient $v$ dominates the diffusion term, so that a high value Péclet number given by $Pe = 10^3$ is obtained. This corresponds to a predominant hyperbolic case. In this case, the analytical solution has a region with a high gradient near to the axes $x = 0$ and $y = 0$, which is similar to the sharp gradient of the boundary value problem of the previous section, except that now it depends on the Péclet number. Outside this zone, a flat region is obtained. The numerical approximation near the boundary is not trivial due to the high spatial variation.

In Table IV, we show the iterated results obtained by Algorithm 1 with the initial shape parameter $c_{\text{ini}}$ =0.01. In the first column, the iteration number is displayed, the adaptivity of the subdomains and nodes are illustrated in the second and third column respectively. The fourth column corresponds to the measurements of the error in the $\|\cdot\|_\infty$ norm.

It can be observed from Table IV that as the number of iterations is increased the $E_{\max}$ decreases giving a better numerical approximation; see fourth column. Simultaneously, the number of subdomains are adequately adapted with respect to the number of nodes, see second column. The initial condition number is $10^9$ and the final condition number is in the interval $[10^{10}, 10^{11}]$. Note in the last three rows in Table IV that there is little variation in the numerical error and the number nodes, a result that could be used as an empirical indicator to stop Algorithm 1.

Recall that unlike the case of Poissons problem, where the $E_{\max}$ diminishes consistently with the number of subdomains, in the case of the convection-diffusion problem the reduccion of the error $E_{\max}$ is considerably slower. This is a consequence of the high Péclet number that we are using, i.e., $Pe = 10^3$.

The gradient of the limit layer of the convection-diffusion problem is much more pronounced that the high sharp region present in the case of Poisson problem. This behavior not only requires a greater number of subdomains, but also implies that its aspect ratios are much greater.

In Fig. 4, the bottom picture displays 8 subdomains whose sizes are nearly $1/8$ of the size of the whole domain. Moreover, in Fig. 6 the final size of the subdomains are extremely narrow near to axes $x = 0$ and $y = 0$, compared with the whole domain. These elements explain why the rate of convergence in Poisson case decreases much more rapidly that in the convection-diffusion problem. However, in this last case, the CPU time is considerably smaller when the domain decomposition method with node refinement is used as compared with the nonadaptive domain decomposition method. This result is consistent with the previous numerical results relative to the Poisson problem.

The cells and subdomains adaptivity at different iterations are illustrated in Fig. 6. In addition, the last two figures correspond to the final iterated numerical solution $\tilde{u}(x, y)$ and the final cell distribution near to the point $(0, 0)$. It can be observed from Fig. 6, that, the cells were refined in the zones where the analytical solution presents a sharp spatial variation. Hence, the function indicator $\eta(x)$ correctly detects the regions where the approximation presents a high spatial variation, generating a denser node distribution and consequently reducing the approximation error. In a similar way, an error reduction occurs in regions with low spatial variation.

Note that the maximum refinement is attained near the point $(0, 0)$ with a value of 13 levels of refinement, which is consistent with the behavior of the analytical solution. On the other hand, in the flat region it was not necessary to refine the cells and the level of refinement keeps a constant value of 5, which correspond to the minimal level in the tree. In addition, the dynamic partition of the data has a concentration of subdomains and a diminution in its sizes near to the axes $x = 0$ and $y = 0$.

As in the previous section, we performed numerical experiments by using the RBFs: $r^4 \log r$, $r^5$ and $r^7$. Unlike for the case of Poisson problem, for the convection-diffusion experiment, we obtained that the differences of the maximum errors between the kernels $r^4 \log r$ and $r^5$, were of nearly the same order of magnitude. Specifically, we obtained that for the RFB $r^4 \log r$, the DD-adaptive Algorithm 1, gave an $E_{\max}$ error of 0.020052, using 8824 nodes and 46 iterations. Simultaneously, for the RBF $r^5$, we obtained an $E_{\max}$ error of 0.016303, using 8836 nodes and 46 iterations. Thus we did not obtain an improvement of the error for this case. These results were obtained for a Péclet number of 500 which means that the gradient of the solution is highly sharp. This bad behavior of the maximum error has also been reported in [13], for a Péclet number of 100 using DD with a Cartesian grid. We would like to emphasize, that for low Péclet numbers, for example 10, the authors in [13] report an excellent improvement in the maximum error for the former RBF, a result which is consistent with the fact that the gradient of the solution is smooth. This is not only consistent with our experience, but also explains why the maximum error decreases in such an important way for the Poisson problem. As in this last case, the kernel $r^7$ produced a blow up in the numerical solution.

Based on the results obtained for the Poisson and the convection-diffusion problem, we can conclude that the proposed Algorithm 1 is an efficient algorithm; the node adaptivity allow us to reduce the number of nodes required to capture the sharp gradient whereas the domain decomposition with dynamic data partition reduces the time required to solve the linear system of equations, keeping low the condition number and letting us handle large scale problems.

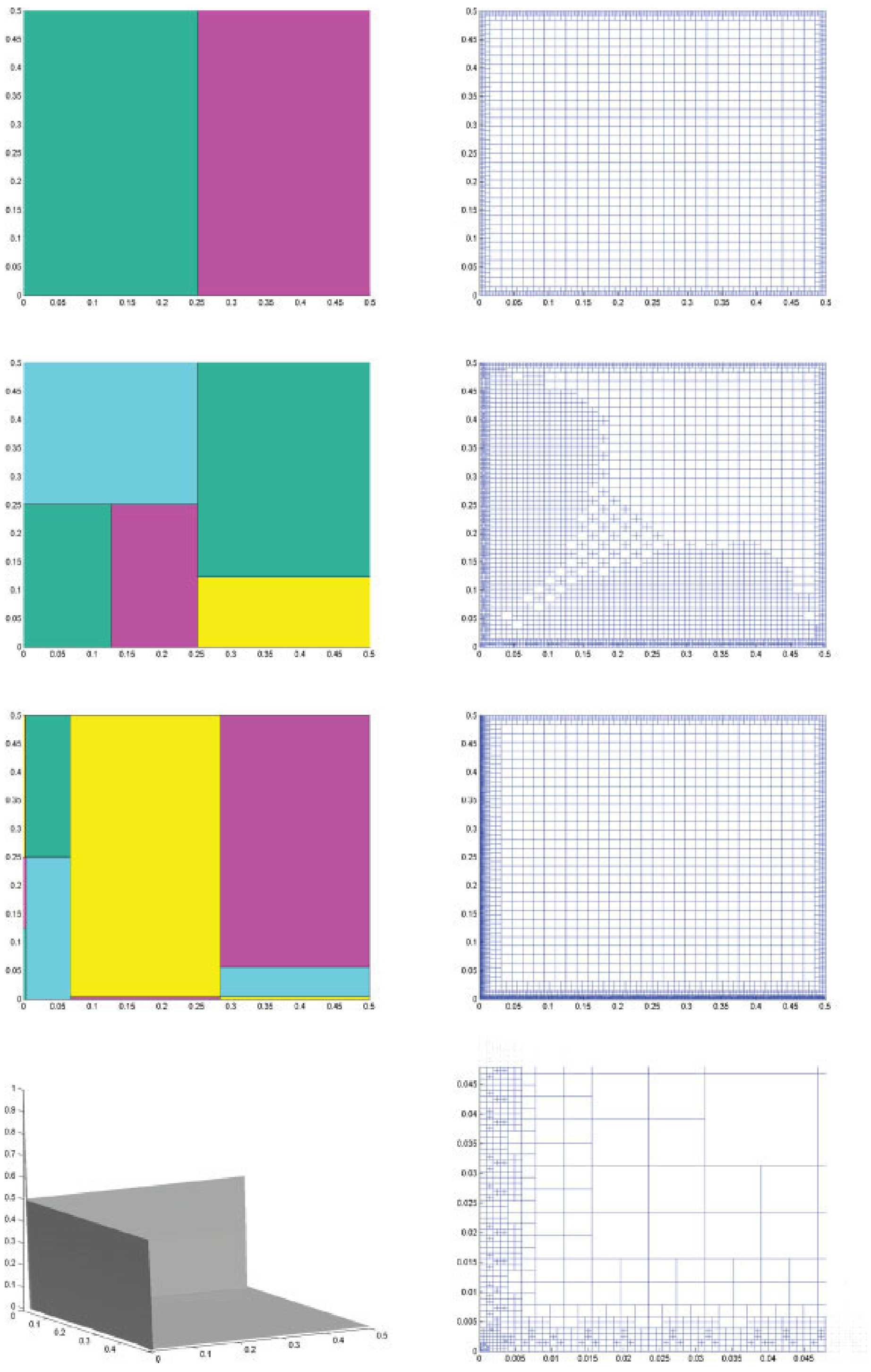

FIG. 6. Dynamically distribution of the nodes and the subdomains for convection-diffusion problem at iterations = {0,1,9}. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

## VII. CONCLUSIONS

In this article, we have introduced a RBF collocation algorithm, based on multiquadric kernels, which combine a domain decomposition technique with a node adaptive algorithm. Our main interest is to solve large PDE problems whose solutions present sharp gradients or oscillations. To this end, a RBF multiplicative Schwarz domain decomposition algorithm is developed to deal with the large size of the problem. Simultaneously, a node adaptive technique is applied to efficiently approximate the high gradient regions of the solution. The combination of both techniques requires that the DD method is able to dynamically adapt the number of subdomains by increasing or decreasing their number where the node density is high or low, respectively. This was achieved by means of a coordinate bisection algorithm. At the same time, the node adaptive method increase the node density in the regions where the solution presents sharp gradients while simultaneously reduces the node density in smooth parts of the solution. The node adaptive algorithm is based on a quadtree structure and local error estimates. An important issue of this algorithm is that the node density smoothly decrease from high density zones to regions with low density. This problem was solved within the node adaptive algorithm by imposing a 2:1 subdivision rule in the quadtree structure. The quadtree structure also plays an important role in the construction of the overlapping regions of each subdomain, as the level in the tree is directly related to the gradient of the solution. We managed to solve the problem by building wider overlapping regions that are placed in sharp parts of the solution than those which are placed in smooth regions. As far as to the authors knowledge, this is the first RBF algorithm reported in literature which combines domain decomposition methods with node adapting techniques.

Two stationary problems in 2D were chosen to numerically analyze this method. A Poisson problem with Dirichlet boundary conditions whose analytical solution presents a sharp gradient was discretized. Also, a convection-diffusion stationary problem with a boundary layer for Péclet numbers up to $10^3$, was studied. For both problems, it was clearly observed that the proposed algorithm effectively adapts both the node density and the subdomain distribution. For the case of the Poisson problem, we observed that as the number of iterations increases the error decays considerably. Also, it was observed that the domain structure and node density increases near the high gradient region of the solution, thus reducing the numerical complexity of the scheme. In the case of convection-diffusion equation, this behavior was also clearly observed, although for Péclet numbers up to $10^3$, the subdomains become highly thin near the boundary layer, thus reducing the rate of the error decay.

It should be observed that the DD-adaptive algorithm reduces the numerical complexity in comparison with the domain decomposition methods without adaptation. It is also worth mentioning that, although through this work a Cartesian subdomain structure has been used, the proposed rule to extend the subdomains and its artificial boundaries, permits us to easily extend this formulation to irregular subdomain structures.

The former results represent an encouraging first step toward the formulation of highly efficient algorithms capable of solving large PDE problems with oscillatory or sharp solutions. A better non-Cartesian structure of the subdomain partition is possible. As mentioned, to obtain global convergence a two-level Schwarz method should be used. Time dependent problems can also be formulated within the frame presented in this article. Further work in these directions is currently being undertaken by the authors.

We express our gratitude to the referees for their many valuable suggestions that improved this article.

## References

1. E. J. Kansa, Multiquadrics—a scattered data approximation scheme with applications to computational fluid dynamics. I. Surface approximations and partial derivative estimates, Comput Math Appl 19 (1990), 127–145.
2. E. J. Kansa, Multiquadrics—a scattered data approximation scheme with applications to computational fluid dynamics. II. Solutions to parabolic, hyperbolic and elliptic partial differential equations, Comput Math Appl 19 (1990), 147–161.
3. G. E. Fasshauer, Solving differential equations by collocation with radial basis functions, A. Le Mehaute, C. Rabut, L. Schumaker, editors, Surface Fitting Multiresolution Methods, Vanderbilt University, Nashville, Tennessee, United States, 1997, pp. 131–138.
4. M. A. Golberg and C. S. Chen, Discrete Projection Methods for Integral Equations, Computational Mechanics Publications, Southampton, Boston, 1997.
5. I. S. Raju, D. R. Phillips, and A. Krishnamurthy, A radial basis function approach in the meshless local petrov-galerkin method for euler-bernoulli beam problems, Comput Mech 34 (2004), 464–474.
6. C. Shu, Differential quadrature and its application in engineering, Springer-Verlag, London, 2000.
7. M. D. Buhmann, Radial basis functions, Cambridge University Press, Cambridge, 2003.
8. G. E. Fasshauer, Meshfree approximation methods with Matlab, World Scientific Publishers, Singapore, 2007.
9. A. H. D. Cheng, M. A. Golberg, E. J. Kansa, and G. Zammito, Exponential convergence and *h-c* multiquadric collocation method for partial differential equations, Numer Methods Partial Differential Eq 19 (2003), 571–594.
10. R. Schaback, Error estimates and condition numbers for radial basis function interpolation, Adv Comput Math 3 (1995), 251–264.
11. A. Iske and J. Levesley, Multilevel scattered data approximation by adaptive domain decomposition, Num Algorithms 39 (2005), 187–198.
12. Z. Wu and Y. C. Hon, Additive Schwarz domain decomposition with radial basis approximation, Int J Appl Math 4 (2000), 81–98.
13. J. Li and C. S. Chen, Some observations on unsymemetric radial basis function collocation methods for convection-diffusion problems, Int J Numer Methods Eng 57 (2003), 1085–1094.
14. L. Ling and E. J. Kansa, Preconditioning for radial basis functions with domain decomposition methods, Math Comput Model 40 (2004), 1413–1427.
15. J. Li and Y. C. Hon, Domain decomposition for radial basis meshless methods, Numer Methods Partial Differential Equ 20 (2004), 450–462.
16. A. S. M. Wong, Y. C. Hon, T. S. Li, S. L. Chung, and E. J. Kansa, Multizone decomposition for simulation of time-dependent problems using the multiquadric scheme, Comput Math Appl 37 (1999), 23–43.
17. J. A. Muñoz-Gómez, P. González-Casanova, and G. Rodríguez-Gómez, Domain decomposition by radial basis functions for time dependent partial differential equations, S. Sahni, editor, Proceedings of the 2nd IASTED International Conference on Advances in Computer Science and Technology, Puerto Vallarta, Mexico, ACTA Press, 2006, pp. 105–109.
18. P. P. Chinchapatnam, K. Djidjeli, and P. B. Nair, Domain decomposition for time-dependent problems using radial based meshless methods, Numer Methods Partial Differential Eq 23 (2007), 38–59.
19. X. Zhou, Y. C. Hon, and J. Li, Overlapping domain decomposition method by radial basis functions, Appl Numer Math 44 (2003), 241–255.
20. Y. Zhang and Y. Tan, Convergence of general meshless schwarz method using radial basis functions, Appl Numer Math 56 (2007), 916–936.

21. J. Behrens, A. Iske, and St. Pohn, Effective node adaption for grid-free semi-lagrangian advection, Discrete modelling and discrete algorithms in continuum mechanics, T. Sonar, I. Thomas, editors, Logos Verlag, Berlin, 2001, pp. 110–119.

22. J. Behrens and A. Iske, Grid-free adaptive semi-lagrangian advection using radial basis functions, Comput Math Appl 43 (2002), 319–327.

23. T. A. Driscoll and A. R. H. Heryudono, Adaptive residual subsampling methods for radial basis function interpolation and collocation problems, Comput Math Appl 53 (2007), 927–939.

24. V. Pereyra, G. Scherer, and P. González-Casanova, Radial function collocation solution of partial differential equations in irregular domains, Int J Comput Sci Math 1 (2007), 28–41.

25. H. Samet, The quadtree and related hierarchical data structures, ACM Comput Surv 16 (1984), 187–260.

26. H. Samet, Hierarchical spatial data structures, Design and implementation of large spatial databases, A. P. Buchmann, O. Gunther, T. R. Smith, Y. F. Wang, editors, Proceedings of the first symposium on Design and implementation of large spatial databases, Santa Barbara, California, United States, 1990, pp. 193–212.

27. M. J. Berger and S. H. Bokhari, A partitioning strategy for nonuniform problems on multiprocessors, IEEE Trans Comput 36 (1987), 570–580.

28. H. D. Simon and S. H. Teng, How good is recursive bisection?, SIAM J Sci Comput 18 (1997), 1436–1445.