Adaptive Node Refinement Collocation Method for Partial Differential Equations

José Antonio Muñoz-Gómez
Ciencias Computacionales
Instituto Nacional de Astrofísica, Óptica y Electrónica
Puebla, Tonantzintla, México
jantionio@inaoep.mx

Pedro González-Casanova
UICA-DGSCA
Universidad Nacional Autónoma de México
D.F., México
pedrogc@dgsca2.unam.mx

Gustavo Rodríguez-Gómez
Ciencias Computacionales
Instituto Nacional de Astrofísica, Óptica y Electrónica
Puebla, Tonantzintla, México
grodrig@inaoep.mx

Abstract

In this work, by using the local node refinement technique proposed in [1, 2], and a quad-tree type algorithm [3, 4], we built a global refinement technique for Kansa’s unsymmetric collocation approach. The proposed scheme is based on a cell by cell data structure, which by using the former local error estimator, iteratively refines the node density in regions with insufficient accuracy. We test our algorithm for steady state partial differential equations in one and two dimensions. By using thin-plate spline kernel functions, we found that the node refinement let us to reduce the approximation error and that the node insertion is only performed in regions where the analytical solution shows a high spatial variation. In addition, we found that the node refinement outperform in accuracy and number of nodes in comparison with the global classical Cartesian h-refinement technique.

Keywords: node adaptive, cell refinement, radial basis functions, partial differential equation, collocation method.

1. Introduction

In this paper we investigate a node refinement technique to improve the spatial resolution of boundary value problems in one and two dimensions. The numerical approximation is built by means of the unsymmetric radial basis function collocation method [5, 6], which has been successfully used in a widely range of practical problems; see [7, 8, 9, 10, 11, 12, 13].

The purpose of local node refinement methods is to add nodes only on those parts of the computational domain where the numerical approximation requires to be more accurate. In fact, this strategy is particularly important when the solution presents regions with different space scale or regions with high gradients.

In this article, by using a quad-tree type technique instead of a Delaunay triangulation, we built an algorithm which successful controls the spatial density distribution and node location of the global refinement algorithm. It is worth to stress, that unlike Delaunay refinement based techniques, this approach makes it possible to be easily extend to three dimensional problems.

The new scheme proposed in this work was applied to two linear partial differential equations problems: a) a 1-d initial boundary problem, and b) a 2-d Poisson problem. Within this context, it was verified that this refinement method efficiently increases the spatial density distribution of nodes in regions where the solution presents sharp gradients and simultaneously keeps enough data points in smooth regions. Moreover, it was verified that the number of nodes required to meet a prescribed error value is considerably reduced when using the purposed node refinement algorithm, with respect to the uniform grid refinement method applied under the same conditions.

The remainder of this paper is organized as follows: In Section 2 we describe the unsymmetric collocation method for a general boundary value problem. The node refinement strategy and the data structure used to improve the numeri-
2. Radial Basis Functions

In this Section, for the purpose of completeness, we present the unsymmetric radial basis functions collocation method for the general steady-state problem in $d$-dimensions.

Consider the following boundary value problem (BVP)

$$Lu = f \quad \text{in} \quad \Omega \subset \mathbb{R}^d, \quad Bu = g \quad \text{on} \quad \partial \Omega, \quad (1)$$

where $L$ is an arbitrary linear differential operator, $\partial \Omega$ denotes the boundary of the domain $\Omega$, and $B$ is a boundary operator, which can be Dirichlet, Neumann or a mixed type. The functions $f, g : \mathbb{R}^d \to \mathbb{R}$ are provided.

In the Kansa approach [5, 6], the problem is discretized by $N$ collocation nodes $\{x_i\}_{i=1}^N \subset \Omega$, which can be divided into interior $\{x_i\}_{i=1}^{N-I}$ nodes and $\{x_i\}_{i=N-I+1}^N \in \partial \Omega$ boundary nodes. In order to obtain the approximate solution $\tilde{u}(x)$ to the exact solution $u(x)$ of the BVP defined by (1), we first define the radial approximation $\tilde{u}(x)$ given by

$$\tilde{u}(x) = \sum_{j=1}^N \lambda_j \phi(||x - x_j||) + p(x), \quad x \in \mathbb{R}^d, \quad (2)$$

where $\lambda_j$ are the unknown coefficients to be determined. Here $\phi(||\cdot||)$, where $||\cdot||$ is the Euclidean norm, is any radial basis function (RBF). The degree of the polynomial $p(x)$ depends on the RBF chosen, for positive definite functions the interpolant does not require a polynomial, for semi-positive RBF it is necessary to incorporate $p(x)$ in order to guarantee the non-singularity. A compressive study of RBF interpolation can be found in the book of Buhmann [14].

Substituting (2) in (1) with $p(x) = 0$, we obtain

$$\sum_{j=1}^N \lambda_j L \phi(||x_i - x_j||) = f(x_i) \quad i = 1, \ldots, N_I, \quad (3)$$

$$\sum_{j=1}^N \lambda_j B \phi(||x_i - x_j||) = g(x_i) \quad i = N_I + 1, \ldots, N, \quad (4)$$

which must be solved for the unknowns $\{\lambda_i\}_{i=1}^N$. The $\lambda$’s are back-substituted into the radial approximation (2) to obtain the approximation of the solution. The linear system is solved by LU factorization.

The most widely used radial basis functions are shown in Table 1, with $r = ||x_i - x_j||$. In our numerical examples we have used the Thin-Plate Splines (TPS) with $m = 4$.

### Table 1. Global Radial Basis Functions

<table>
<thead>
<tr>
<th>RBF</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiquadric (MQ)</td>
<td>$\phi(r, c) = \sqrt{r^2 + c^2}$</td>
</tr>
<tr>
<td>Inverse Multiquadric (IMQ)</td>
<td>$\phi(r, c) = 1/\sqrt{r^2 + c^2}$</td>
</tr>
<tr>
<td>Gaussian (GA)</td>
<td>$\phi(r, c) = e^{-cr^2}$</td>
</tr>
<tr>
<td>Thin-Plate Splines (TPS)</td>
<td>$r^m \log r, \ m = 2, 4, 6, \ldots$</td>
</tr>
<tr>
<td>Smooth Splines (SS)</td>
<td>$r^m, \ m = 1, 3, 5, \ldots$</td>
</tr>
</tbody>
</table>

which avoid the complexity introduced for the selection of the shape parameter $c$ used in MQ, IMQ and GA.

To the best authors knowledge the non-singularity of the linear system (3,4) obtained by the Kansa approach is an open question. For the interpolation problem using TPS it is necessary to include a linear polynomial; see Section 3.1. In our case for the partial differential equation (PDE) approximation with TPS we do not include a polynomial, this fact has been reported numerically successfully [15, 16, 17].

3. Adaptive Node Refinement

In the first part of this Section we describe the error indicator function to mark the nodes that will be refined. Then, we introduce the data structure used to perform the node refinement scheme for non-equally spaced data.

Although the field of adaptive node methods for RBF is still a young area of research, significant contributions have already appeared in the literature. Behrens et al. [1, 2], have formulated an adaptive strategy based on local RBF interpolants in two dimensions. Their technique has been successfully applied, in a semi-Lagrangian context, to linear evolutionary PDE’s. In [18, 19], an adaptive meshfree RBF algorithm for nonlinear scalar conservation laws was developed and successfully applied to the inviscid Burger equation. Although this method is formulated in one dimension, the author points how to extend his technique to two dimensions. Other techniques [20, 21, 22], mainly in one dimension, have been also developed for different RBF. As previously pointed out, in this contribution we construct an algorithm which combines Behrens et al. refinement method with a quad-tree type algorithm in two dimensions.

3.1 Error Indication

In this Section, we depict the error indicator function to mark the nodes that will be refined. Then, we introduce the data structure used to perform the node refinement scheme for non-equally spaced data.

The error indicator function to mark the nodes that will be refined is defined by

$$\eta(x) = \sum_{i=1}^N \lambda_i \phi(||x - x_i||)$$

where $\lambda_i$ are the unknown coefficients to be determined. Here $\phi(||\cdot||)$, where $||\cdot||$ is the Euclidean norm, is any radial basis function (RBF). The degree of the polynomial $p(x)$ depends on the RBF chosen, for positive definite functions the interpolant does not require a polynomial, for semi-positive RBF it is necessary to incorporate $p(x)$ in order to guarantee the non-singularity. A compressive study of RBF interpolation can be found in the book of Buhmann [14].

Substituting (2) in (1) with $p(x) = 0$, we obtain

$$\sum_{j=1}^N \lambda_j L \phi(||x_i - x_j||) = f(x_i) \quad i = 1, \ldots, N_I, \quad (3)$$

$$\sum_{j=1}^N \lambda_j B \phi(||x_i - x_j||) = g(x_i) \quad i = N_I + 1, \ldots, N, \quad (4)$$

which must be solved for the unknowns $\{\lambda_i\}_{i=1}^N$. The $\lambda$’s are back-substituted into the radial approximation (2) to obtain the approximation of the solution. The linear system is solved by LU factorization.

The most widely used radial basis functions are shown in Table 1, with $r = ||x_i - x_j||$. In our numerical examples we have used the Thin-Plate Splines (TPS) with $m = 4$,...
Let $\Xi$ the current node set from the initial knots distribution. From each node $x \in \Xi$ we select a set of neighboring nodes $N_x \setminus x \in \Xi$. We construct a local interpolant $I\hat{u}(x)$ based on $N_x \setminus x$, using the equation (2) plus a polynomial of degree 1 with a TPS as kernel function.

Based on the numerical approximation $\hat{u}(x)$ and the local interpolant $I\hat{u}(x)$ we define the error indicator

$$\eta(x) = |\hat{u}(x) - I\hat{u}(x)|,$$

which assign a significance value $\eta(x)$ to each node $x \in \Xi$. The value of $\eta(x)$ is small when the surroundings values of $\hat{u}(x)$ are similar or when can be represented by a linear function. On the other hand, a high value $\eta(x)$ commonly indicates that the numerical approximation around of $\hat{u}(x)$ has a sharp variation, which generally correspond to regions with high gradient.

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

**Definition A** Let $\eta^* = \max \eta(x)$ be for all $x$, and let $\theta_{\text{ref}} > 0$ a threshold value. We say that a node $x \in \Xi$ is flagged to be refined if $\eta(x) > \theta_{\text{ref}} \cdot \eta^*$.

This definition can be understood as: only the nodes which has an error upper a threshold value, based on the percentage $\theta_{\text{ref}} \in (0, 1)$, are marked to be refined. The refinement strategy illustrated above, is applied in an iterative refinement algorithm, where at each step we solve the analyzed PDE problem, and with the numerical approximation $\hat{u}(x)$ obtained, we perform the refinement scheme. This algorithm is described in detail in the next Section.

### 3.2 Adaptive Cells

The main difference with respect to [1, 2] is the data structure used to localizes and insert the nodes. In our work we does not employ a Delaunay triangulation or his dual Voronoi to search and insert the nodes respectively.

The domain $\Omega$ is divided in a non overlapping cell by cell structure as shown in Figure 1, the center of each cell contains a node; more details in [3, 4]. Throughout this work the terms of cell refinement and node refinement are used indifferently. Based on the local error indicator $\eta(x)$ we flagged the cells to be refined. For each cell marked for refinement, four nodes are inserted. This strategy allows us to obtain a greater density of nodes in the zones with high gradient.

The data structure used assigns to each cell a pointer to its natural 4-neighbors: up, down, right and left. In order to construct the RBF local interpolator $I\hat{u}(x)$, the set of neighbors nodes $N_x$ of a point $x_i$ is obtained from its naturals 4-neighbors. In addition, the neighboring nodes in the diagonals are included: up-right, up-left, down-right and down-left.

![Figure 1. Illustration of the cell refinement in two dimensions, it is observed that the neighboring cells are refined by relation 2:1](image)

Keeping explicitly the reference to its natural 4-neighbors by each cell structure, we are increasing in 32 bytes the cost of storage in memory. Nevertheless, the complexity to search his neighboring nodes is $O(1)$.

The cell structure described above can be applied to one dimension stationary problems. Where, the node refinement strategy consists in the insertion of two nodes by each node marked to be refined. That is to say, let $x_0 < x_1 < \cdots < x_N$ and let $x_i$ the flagged node to be refined. For each flagged node, it is inserted a node to the left and right of $x_i$:

$$x_l = x_i + \frac{1}{2}(x_{i+1} - x_i),$$
$$x_r = x_{i-1} + \frac{1}{2}(x_i - x_{i-1}),$$

and $x_i$ is eliminated.

It should be observed that the utilized data structure can be extended to three or more dimensions, and together with the error indicator, consist in a truly meshfree scheme for non-equally spaced data for higher dimensions.

### 4 General Scheme

In this Section we integrate the error indicator and the scheme based on cells, in order to construct a general algorithm that allows us to solve numerically problems of initial boundary value with local spatial refinement using radial basis functions.

The sketch of the algorithm is given by:

- Initialize the coarse node distribution
- Iterate until convergence
  - Solve the PDE problem with RBF
  - Perform the adaptive refinement scheme

In this work the convergence is reached when the root mean square ($l_2$) or the infinity error ($l_\infty$) is lower than a given fixed value $\tau$

$$l_2 = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\tilde{u}_i - u_i)^2},$$

$$l_\infty = \max |\tilde{u}_i - u_i|, \quad \text{for all } i,$$

where $\tilde{u}_i = \tilde{u}(x_i)$ and $u_i = u(x_i)$, with $x_i \in \mathbb{R}^d$, represent the numerical and exact solution respectively.

The proposed algorithm is composed of two blocks: numerical solution of the PDE for non-equally spaced nodes, and the scheme of local refinement in 2-d. To the best of authors’ knowledge, the second block described has not been used in the context of radial basis functions. This is the first step to find out a general scheme of a block structured refinement for 3-d problems with complex geometries.

For a simple iteration of the algorithm, the analyzed PDE is resolved by the unsymmetric collocation method. In this way, we obtain the approximate solution $\tilde{u}(x)$ for every node of the domain $\Omega$; see Section 2. Based on the numerical solution obtained, it is built the error indicator $\eta(x)$ in order to determine the nodes that will be refined; see Section 3.1. Finally, the flagged nodes are refined employing the cell structure described in Section 3.2.

Each module described above are in different executable files and the interface is made by means of binary files. This modularity permits us to extend the functionality of the algorithm proposed without having to change all the programs. The software is available upon request to the first author.

5. Results and Discussion

The aim of this section is to obtain the numerical solution of a boundary value problems with the node refinement scheme proposed. For this purpose we analyze two boundary value problems in one and two dimensions. In addition, we investigate the accuracy of the node refinement scheme in comparison with a uniform grid distribution.

5.1 Example 1-d

Consider the following partial differential equation in one dimension

$$u_{xx}(x) = f(x), \quad x \in \Omega \subset \mathbb{R},$$

where $\Omega=[0,1]$, with Dirichlet boundary conditions

$$u(x) = g(x), \quad x \in \partial \Omega.$$

The subindexes in $u_{xx}$ denote the second partial derivative with respect to $x$. The analytic solution is given by

$$u(x) = tanh(\eta(x - 0.5)),$$

with $\eta > 0$, the function $f(x)$ is expressed as

$$f(x) = -2\eta u(x)(\eta - \eta u(x)^2).$$

The shape parameter $\eta$, allows us to control the degree of smoothness of the analytic solution. Increasing the value of the coefficient $\eta$ it is obtained a discontinuity in 0.5.

Consider a distribution of 34 equally spaced nodes, with a threshold refinement value $\theta_{ref} = 0.1$, the stop criterion is established when $l_\infty < 10 \times 10^{-3}$. We defined the coefficient $\eta=500$, from which we obtain a high gradient in the analytic solution without arriving at the discontinuity.

Table 2 shows the decreasing of the error as the refinement algorithm is iterated; see Section 4.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$N$</th>
<th>$l_\infty$</th>
<th>$l_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>34</td>
<td>0.968643</td>
<td>0.559801</td>
</tr>
<tr>
<td>1</td>
<td>45</td>
<td>0.912644</td>
<td>0.533539</td>
</tr>
<tr>
<td>2</td>
<td>74</td>
<td>0.172233</td>
<td>0.084677</td>
</tr>
<tr>
<td>3</td>
<td>98</td>
<td>0.415363</td>
<td>0.249337</td>
</tr>
<tr>
<td>4</td>
<td>158</td>
<td>0.014681</td>
<td>0.008472</td>
</tr>
<tr>
<td>5</td>
<td>172</td>
<td>0.001712</td>
<td>0.000305</td>
</tr>
<tr>
<td>6</td>
<td>194</td>
<td>0.000352</td>
<td>0.000211</td>
</tr>
</tbody>
</table>

Table 2. Decreasing of the error vs Iterations.

Figure 2 shows the reconstruction of the numerical approximation with the node refinement scheme. It can be observed from this graph, that the zone of high gradient is well captured.

The distribution of the nodes of Figure 2 is showed in Figure 3. From the last picture, it can be observed that near the point 0.5 we obtain a greater nodes density, and when we moved away of the point 0.5 the node distribution is diminishes. This gradual diminution of nodes correspond to the imposed restriction 2:1 in the algorithm. In the flat zones of the analytical solution, we do not required of many nodes to obtain a good numerical approximation.

5.1.1 Adaptive vs Non-Adaptive

In this experiment, we compare the adaptive node refinement scheme vs the uniform grid distribution to meet the same error.
Figure 2. Reconstruction of $u(x)$ after the node refinement algorithm.

Figure 3. Histogram of the nodes distribution.

The test was carry out using a coefficient $\eta=1000$, with and without node refinement. Figure 4 shows two graphs in a loglog scale, in the x-axis we show the number of nodes and in y-axis correspond to the root mean square error. The global refinement consist in successive increments in the number of nodes in the interval $[0, 1]$, where the nodes belongs to the set $\{100, 200, \ldots, 1400\}$. As we can see from Figure 4, for to reach the same root mean square error, the node refinement require less nodes in comparison with the global refinement. Therefore, the node refinement improve the computational throughput.

In Table 3 shown the global $h$-refinement and local $h$-refinement to reach the same $l_2$-error. In order to meet the same $l_2$-error, see Table 3, the local $h$-refinement scheme requires an 82% less nodes in comparison with the required nodes of the global $h$-refinement technique.

In the third column of Table 3 it is observed that the norm $l_\infty$ is approximately one order of magnitude better than the obtained with the global refinement. In order to achieve a similar error with the global refinement, we require of 2700 nodes distributed uniformly, obtaining an error $l_\infty=8.281\text{e}-04$. This is approximately to 10 times more than the number of nodes required with the local adaptive technique. This difference in the number of nodes is dramatical since the asymmetric collocation method generates a dense matrix, and when is solving with a direct methods the algorithm complexity is $O(N^3)$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>N</th>
<th>$l_\infty$</th>
<th>$l_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local $h$-refinement</td>
<td>249</td>
<td>8.151e-04</td>
<td>4.086e-04</td>
</tr>
<tr>
<td>Global $h$-refinement</td>
<td>1400</td>
<td>3.546e-03</td>
<td>4.160e-04</td>
</tr>
</tbody>
</table>

Table 3. Comparison between global $h$-refinement and local $h$-refinement.

In order to meet the same $l_2$-error, see Table 3, the local $h$-refinement scheme requires an 82% less nodes in comparison with the required nodes of the global $h$-refinement technique.

5.2 Example 2-d

Consider the following Poisson problem in two dimensions

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \text{ in } \Omega \subset \mathbb{R}^2,$$

(10)
\[ u = u(x, y), \] together with Dirichlet boundary conditions

\[ u = g \text{ on } \partial \Omega, \quad (11) \]

where \( \Omega = [0,1]^2 \). The functions \( f, g : \mathbb{R}^2 \rightarrow \mathbb{R} \) are determined using the analytical solution

\[ u(x, y) = e^{-50x} + e^{-50y}. \quad (12) \]

Figure 5 shows the shape of the analytical solution. As we can observe from this graph, near to the axes \( x = 0 \) and \( y = 0 \), the analytical solution have a region with a high gradient, similar to a boundary layer problem. Outside this zone a flat region is obtained. The determination of a good numerical approximation near the boundary is not a trivial problem, since the high spatial variation.

The stop criterion is established when \( l_\infty < 5 \times 10^{-3} \), with an initial node distribution of \( N = 324 \) equally spaced nodes, and \( \theta_{ref} = 0.3 \) as a threshold value for the node refinement scheme.

In Table 4 is showed the diminishing of the error in the numerical approximation as the algorithm of node refinement is iterated; see Section 4. In the first column the iteration number is showed, the number of nodes is displayed in the second column. The third and fourth column correspond to the measurements of the error using the norms \( l_2 \) and \( l_\infty \) respectively.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>N</th>
<th>( l_2 )</th>
<th>( l_\infty )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>324</td>
<td>0.043148</td>
<td>0.141574</td>
</tr>
<tr>
<td>2</td>
<td>460</td>
<td>0.021939</td>
<td>0.076647</td>
</tr>
<tr>
<td>4</td>
<td>803</td>
<td>0.008232</td>
<td>0.041500</td>
</tr>
<tr>
<td>6</td>
<td>1111</td>
<td>0.003424</td>
<td>0.034594</td>
</tr>
<tr>
<td>8</td>
<td>1463</td>
<td>0.002493</td>
<td>0.007224</td>
</tr>
<tr>
<td>10</td>
<td>2646</td>
<td>0.001170</td>
<td>0.006585</td>
</tr>
<tr>
<td>12</td>
<td>3776</td>
<td>0.000553</td>
<td>0.004722</td>
</tr>
</tbody>
</table>

Table 4. Decreasing of the error vs. iterations for Poisson problem.

It is observed from the third and fourth column of Table 4, that as the number of iterations is increased, both types of error are diminished. Reaching the objective to reduce the approximation error by means of a local scheme of node refinement.

It can be observed from Figure 6, that the cells where refined in an incremental way. The refined zones are near the axes \( x = 0 \) and \( y = 0 \), which corresponds to the zones in where the greater spatial variation in the analytic solution exists. Therefore, the indicator function \( \eta(x) \) permit us to detect regions where exists a high spatial variation, conforming a best spatial resolution in that regions and obtaining a reduction in the approximation error.

As it was observed, from Table 4 it can be deduced that it is possible to reduce the approximation error when the iteration is increased. However, for further iterations, iterations \( > 12 \), the a-posterior error does not appears in the zone of the high gradient. In this case, the maximum error appears near the axis \( x = 1 \) and \( y = 1 \). This problem has not been solved in the present work and will be the subject of future research.

In order to compare the results obtained by using node adaptive versus non-adaptive techniques, we solved the
same Poisson problem with a uniform grid distribution. Let \( N = 3844 \) be equally spaced nodes, which are divided into \( N_I = 3600 \) interior nodes and \( N_F = 244 \) boundary nodes. In this case, the obtained error is \( l_2=0.015945 \) and \( l_\infty=0.031331 \).

Recall that the number of nodes used in the Cartesian grid case, is similar to the one obtained as a result of the adaptive nodes algorithm; see Table 4. Also note that the error in the adaptive case is \( l_2=0.000553 \) and \( l_\infty=0.004722 \). From this comparison it turns out that the local node refinement strategy improves the numerical approximation by simultaneously reducing the cost of a global mesh refinement.

6. Conclusions

In this work we proposed an adaptive node refinement scheme for steady state partial differential equations by using the unsymmetric radial basis function collocation method. The local refinement technique is based on the work developed on the works \([1, 2, 3, 4]\). The scheme proposed it is a truly meshfree method for node adaptivity. The use of a quad-tree algorithm, instead of a Delaunay technique as purposed by Behrens et al., makes it possible to generalize our approach to three dimensional problems. The node adaptivity was proved to be successful for two boundary value problems in one and two dimensions, this approach showed that the node refinement permit us to capture regions with high gradients, without to insert additional nodes in the regions of low gradients. The computational 1-d experiments showed that is possible to achieve a similar root mean square error to the accomplished with the classical Cartesian \( h \)-refinement scheme, but employing a fewer number of nodes. Thus, we could improve the computational effort.

The bottleneck of the proposed approach is in the LU factorization used to solve the linear system of equations, which is obtained by the discretization of the partial differential equations. A way to reduce the computational effort is to use a refinement method based on the non-homogeneous part of the PDE. Preliminary results have shown favorable outcomes. The above strategy is a first step to obtain a general scheme for a block structured refinement in 3-d.

7. Acknowledgements

The authors wish to thank the support provide from INAOE and UNAM to develop this research. The first author extend his thanks to the Consejo Nacional de Ciencia y Tecnología of México (CONACYT) for supporting this work under grant 145052. This work was partially supported by the CONACYT under the project CONACYT-2002-C01-4022.

References


