A Multigrid-DDM Schur Elliptic Equation Solver in Unstructured Meshes

Guilherme Galante[1], Rogerio L. Rizzi[2] and Tiaraju A. Diverio[1]

[1] PPGC, Instituto de Informática
Universidade Federal do Rio Grande do Sul
CP 15064, 91501-970, Porto Alegre, RS, Brazil
[2] Centro de Ciências Exatas e Tecnológicas
Universidade Estadual do Oeste do Paraná
Rua Universitária, 2069, 85801-110, Cascavel, PR, Brazil
ggalante@inf.ufrgs.br

Abstract. This work shows the parallel Multigrid-MDD Schur method for the solution of elliptic equations. In the proposed method the solution is obtained by a multigrid method parallelized by domain decomposition techniques, more specifically by the Schur complement method. It is also shown some issues related to the generation and partitioning of the mesh hierarchy. In the case study, we used the 2D heat diffusion equation in unstructured meshes. The implementations was developed to explore parallelism in clusters, using message passing.

1 Introduction

The large linear equations systems, that raises of the discretization of partial differential equations (PDE) in technological and scientific problems, require efficient solution. The use of direct methods is inadequate to solve these systems, once a time that do not use the advantage of the coefficients sparsity, making this approach difficult, by storage problems and for the dependence of operations that difficult its parallelization.

The iterative algorithms, however, use only the matrix as operator to iteratively construct a convergent sequence of solutions. And, in contrast of the direct methods, are very used in the resolution of sparses and large equations systems, due its storage optimization potential and computational efficiency. Currently parallel solutions by iterative methods, combine preconditioned Krylov sub-spaces methods as local solver, with domain decomposition methods [1][16].

Other approach used in the equation systems resolution are the multigrid methods. Multigrid methods originated in the 1960s with the work of Fedorenko and Bakhvalov. They were further developed in the 1970s by Brandt, and are now the preferred methods for solving elliptic partial differential equations [19]. The advantage of multigrid is the speed - multigrid algorithms only require order N operations to solve elliptic equations, where N is the number of mesh points.

[1] Candidate to the best student paper award

Since 1980, the number of publications related to the multigrid methods had increased substantially, and currently there are more than 3600 references that can be found in the MGNET (*www.mgnet.org*), who is the official repository of multigrid methods [9].

In this paper, we describe our solver, MG-Schur, that solves elliptic equations using multigrid methods parallelized by domain decomposition methods. The idea of combining multigrid and domain decomposition methods is not new [3][8], although there are a number of features of our code that distinguish it from the discussions we have seen. In particular, we use the Full Multigrid V cycle combinated with the Schur complement domain decomposition method to explore the parallelism. Moreover, our method uses only Krylov space iterative methods, more specifically the GMRES, instead classical iterative solvers, like Gauss-Seidel, normally used in the multigrid approaches. This choice was based considering that we have nonsymmetric systems, and the GMRES is the more appropriate method to solve this class of systems.

This paper is structured as follows: in section 2 generation and the partitioning of meshes are discussed, and a tool *MGTool* is presented; in section 3 a overview of multigrid is presented; in section 4 the proposed parallel multigrid method is presented; in section 5 the study case is described; the section 6 summarizes the results obtained and in section 7 some conclusions and future works are presented.

2 Mesh Generation and Partitioning

In general, problems that use simulation are based on mathematical models that are expressed through Partial Differential Equations (PDE). This PDE, generally, does not have known analytical solution, being necessary the use of discretization and approximation methods, as finite volume or finite element, so that they can be numerically solved.

With the use of numerical techniques of solution of PDE, the region of the domain Ω is not treated as continuous, but like a discrete and finite set of points or subdomains in which the variables are calculated. This discrete set of points or subdomains constitutes the mesh.

In this work the domain discretization is made using unstructured triangular meshes. Unstructured meshes conciliate good representation of the computational domain, since diverse problems are defined in domains with irregular geometry that not always are appropriately discretized by structured meshes. More, specifically we used a special type of meshes, called unstructured orthogonal mesh. It is assumed that within each triangle, there exists a point (hereafter called center) such that the segment joining the centers of two adjacent triangles and the side shared by the two triangles have a nonempty intersection and are orthogonal to each other [6]. The use of this type of mesh simplifies some issues in the PDE discretization, when using finite volumes method.

In the multigrid methods, is necessary create a hierarchy of meshes. In contrast of the work of Chan and Smith [7], we consider the multilevel mesh hierarchy generation starting from a coarse mesh.

The initial coarse mesh is generated by the software *Easymesh* [13]. The *Easymesh* is a program that generates two dimensional, unstructured, Delaunay and constrained Delaunay triangulations in general domains. The mesh quality is achieved by the use of smoothing algorithms.

Once the coarser mesh was generated, we can create the mesh hierarchy. For the generation of the mesh levels we have implemented a tool called *MGTool*. The tool takes the data generated by *Easymesh* and generates the mesh hierarchy, as shown in Fig. 1.

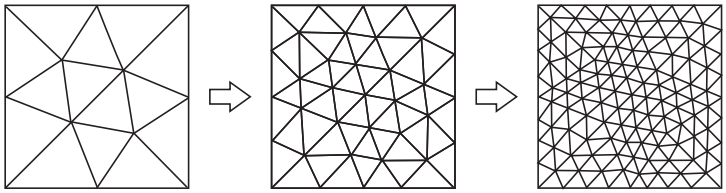
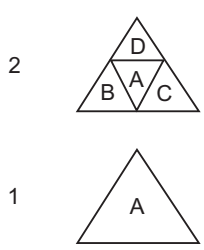


Fig. 1. Example of 3-level mesh hierarchy.

For the refinement of the meshes we adopted a strategy known in literature as h-refinement, characterized for the subdivision of the cells of the domain [10], and is similar to creation of the mesh hierarchy in structured meshes. Refined meshes are created through successive subdivisions of the triangles of the coarse mesh in four subtriangles. In the example shown in Fig. 2, the triangle 1A is refined into four triangles, 2A, 2B, 2C and 2D. Then, the triangle 1A is the "parent" of the triangles 2A, 2B, 2C and 2D, and the triangles 2A, 2B, 2C and 2D are the "children" of triangle 1A.



Cell	Parent	Children			
2A	1A	-	-	-	-
2B	1A	-	-	-	-
2C	1A	-	-	-	-
2D	1A	-	-	-	-

Cell	Parent	Children			
1A	-	2A	2B	2C	2D

Fig. 2. Example of a 2-level mesh hierarchy. The numbers on the left denote the resolution level, and the letters label triangles of the mesh. On the right side, the table that describes the relationship between adjacent levels.

Besides generating the mesh hierarchy, the *MGTool* generate the matrices to each mesh level and also creates information necessary to the interpolation and restriction operators, explained later in Section 3. This informations describes the relationship between adjacent levels, indicating how the data will be transfered from the "parents" to the "children", and vice versa.

To solve the problems in parallel, is necessary that the mesh be partitioned and distributed among the available processors. The partitioning must distribute the workload to each processor in a proportional way. Moreover, due the necessity of information exchange, the partitioning must be made in order to reduce the boundaries among subdomains, and consequently the communications among the processors, since the information exchange is restricted to the boundaries [14]. For the mesh partitioning *MGTool* uses the *METIS* package [12].

The *MGTool* performs the partitioning in the coarser mesh and replicates to the other levels. This approach prevents the load unbalancing, and is advantageous for simulations that run on a large number of processors [8].

3 Multigrid

Multigrid methods is a group of algorithms and techniques that efficiently solve equations systems through the acceleration of the convergence of iterative methods. Basically, the methods multigrid consider a sequence of meshes for the solution of equations system.

To solve a equation system with a multigrid methods in a refined mesh of size N, we introduce meshes of size $N/4$, $N/16$, etc., covering the computational domain. On each grid an associated equation is solved. The solution on the coarse meshes quickly captures the long wavelength features of the solution, and solving on the fine mesh captures the short wavelength features.

The multigrid algorithms are based on three central ideas [18]:

1. communication among meshes;
2. nested iterations;
3. and coarse mesh correction.

A meshes in the multigrid hierarchy communicates with the adjacent ones through restriction and interpolation operators. Restriction takes data on a mesh and restricts it to the next coarsest mesh, defined by:

$$I_N^{N/4} : N \rightarrow N/4$$

Interpolation takes data on a mesh and interpolates it onto the next finest mesh. The operator is defined by:

$$I_{N/4}^{N} : N/4 \rightarrow N$$

In the technique of nested iterations, the objective is find a better initial guess for the solution using iterations in coarse meshes. The coarse meshes have a lower number of variables, and consequently, the computational cost of the

iteration is reduced, in relation to an iteration in the most refined mesh. Then, one better initial guess for $Ax = b$ can be given using the coarsest mesh. An example is show in Fig. 3

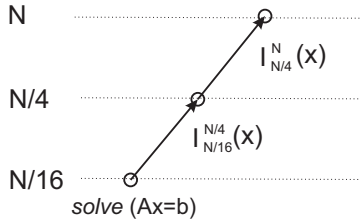


Fig. 3. Nested iterations. In this example, the process starts in the $N/16$ mesh that obtains a initial guess to $N/4$, and $N/4$ compute a initial guess to N.

This strategy does not guarantee that the solution solution in N does not contains soft error components (low frequency). The usage of the correction of error in the coarse meshes prevents this limitation. Iterating in the fine mesh until the errors have been removed, the residual equation is iterated equation in the coarse mesh to get an approximation of the error. Then, is interpolated to the fine mesh, where the first guess is corrected. The correction using the residual equation to iterate the error is shown in Fig. 4.

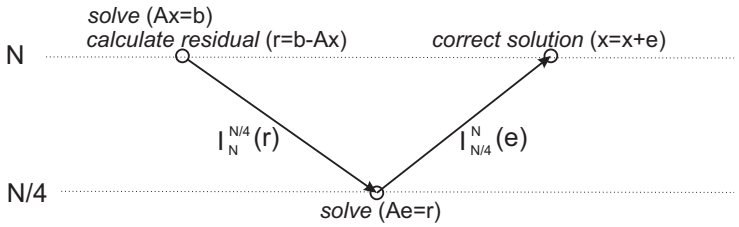


Fig. 4. Coarse grid correction.

Using combinations between the strategy of coarse mesh correction and the use of nested iterations, a family of multigrid methods can be defined [19].

3.1 Full Multigrid V

By the distinct combinations is possible to generate different algorithms that are known in technique literature. A summary of these different approaches can be found in [18][19]. In this work the *full multigrid V*, or simply FMV, is used.

The FMV strategy initiates in the coarsest mesh, for the acquisition of an initial solution with low computational cost for the superior levels. Then, the

levels number is incremented by one, and a coarse mesh correction is performed. This process is repeated until all the levels are involved. In the Fig. 5, the FMV scheme is shown, where the black points represents the operations (linear system solution, residual and error calculations), and the arrows represents the transference of information between the mesh levels.

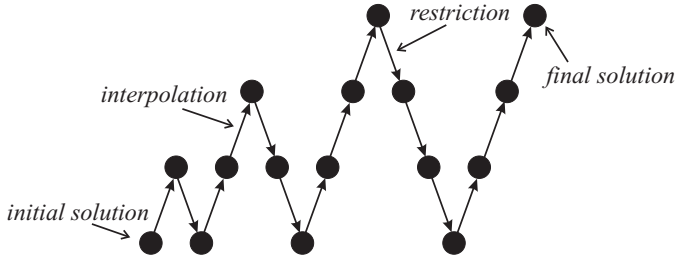


Fig. 5. Full multigrid V.

The theoretical results and numerical experiments show that these methods are efficient and can be applied to an huge types of problems on the scope of the scientific computing. The literature shows the generality of the method, being possible to use it in distinct types of meshes, as well the different discretization methods [4].

4 Parallel MG-Schur

The proposed method uses the domain decomposition approach to explore the parallelism em clusters. Domain decomposition methods (DDM) denotes a set of mathematical, numerical and computational methods and techniques to solve problems in parallel computers.

A DDM is characterized by the division of the computational domain, that is partitioned in subdomains using partitioning algorithms. The global solution of the problem is obtained by the combination of subproblems that are locally solved. Each processor is responsible for finding the local solution of one or more subdomains [17]. In parallel solutions by domain decomposition, the subdomains can be almost treated independently. Therefore, such methods are attractive for distributed memory environments.

The parallel multigrid method was obtained by the parallelization of the FMV operations. The solution smoothing is done in parallel by the Schur complement method. The interpolation, restriction and residual calculations, needed in the coarse mesh correction, also are made in parallel. In the section 4.1 and 4.2 we show the parallelization process of each operation. Thus, all operations are executed in parallel, where each processor is responsible by one subdomain. A ilustration of the solution process can be observed in Fig. 6.

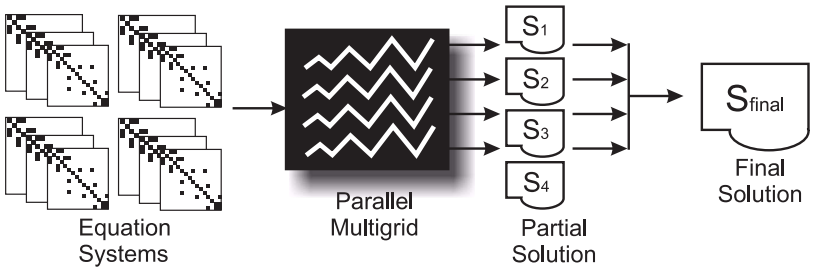


Fig. 6. Solution process

4.1 DDM Parallel Smoother

In this work we used the Schur complement method. In the Schur complement method, the domain is partitioned and the the cells are reordered such that the interface points are listed last after the interior points.

With this local ordering, each local vector of unknows x_i is split into two parts: the subvector u_i of internal vector components followed by the subvector y_i of local interface vector components. The right-hand side b_i is conformally split into the subvectors f_i and g_i. When partitioned according to this splitting, the local matrix A_i, residing in the processor i has the form

$$A_i = \begin{bmatrix} B_i & E_i \\ F_i & C_i \end{bmatrix},$$

so the local equations can be written as:

$$\begin{aligned} B_i u_i + E_i y_i &= f_i \\ F_i u_i + C_i y_i + \sum_{j \in N_i} E_{ij} y_j &= g_i \end{aligned}$$

where N_i is the set of indices for subdomains that are the neighbors to the subdomain i, and the term $E_{ij} y_i$ the contribution to the local equation from the neighboring subdomain j.

Eliminating the variable u in the first equation:

$$u_i = B_i^{-1}(f_i - E_i y_i) \tag{1}$$

and substituting u in the second equation:

$$S_i y_i + \sum_{j \in N_i} E_{ij} y_j = g_i - F_i B_i^{-1} f_i \tag{2}$$

where $S_i = C_i - F_i B_i^{-1} E_i$, and is the local Schur complement.

With this formulation, each processor needs to know the processors with which it must communicate and the list of interface points. Thus, the solution y_i in the interface is obtained, and are used to find the internal variables u_i. For more details about the Schur complement method, see [15].

4.2 FMV operations

The other operations required in the FMV method is now discussed. As well the solver, the restriction, interpolation and residual operations are also calculated in parallel.

For the data transfer operators, we choose $I_N^{N/4}$ to be the full-weighted restriction operator [18], and the interpolation $I_{N/4}^N$ is given by $x_N = \frac{1}{4}\sum_{i=1}^{4} x_{N/4}$, where $x_{N/4}$ is the corresponding values in the coarse mesh. In this operations, each processor is responsible only for its domain. None communication is necessary because no external data are needed.

The residual calculation is necessary in the coarse mesh correction. The residual equation is given by:

$$r_i = b_i - A_i x_i$$

however, as the matrix and the vectors was splitted to the Schur complement method, the residual vector must be splitted in two parts t_i and v_i, where t_i is related to internal components and v_i is related to interface components. Thus, the residual equation must be written as follow:

$$\begin{aligned} t_i &= f_i - B_i u_i + E_i y_i \\ v_i &= g_i - F_i u_i + C_i y_i + \sum_{j \in N_i} E_{ij} y_j \end{aligned}$$

As seen before, the term $E_{ij}y_j$ is the contribution of the neighboring subdomains, therefore, the domain must communicates with its neighbors to receive the required data.

5 Study Case: heat diffusion equation

For numerical and computational experiments, it was used the discretized heat diffusion PDE through the finite volume method under triangular unstructured meshes. The finite volume method subdivides the computational domain in not overlapped subdomains the called finite volumes, that here are the triangles of the mesh.

The finite volume method assures the method quality solution basing the approaches in the conservation principles. For example, the principle of the mass conservation affirms that the mass cannot be created nor be destroyed; if the flow for inside of a region exceeding that one that leaves, the mass will be accumulated inside of this. This approach used here can be implemented by integration of the differential equations, assuring local conservation and, consequently, global [2].

Considering the equation:

$$\frac{\partial T}{\partial t} = \mu \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \tag{3}$$

that model the temperature diffusion, where μ is the diffusion constant.

Integrating the equation (3) in time and space:

$$\int_{\Omega}\int_{t}\frac{\partial T}{\partial t}d\Omega dt = \int_{\Omega}\int_{t}\mu\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)d\Omega dt \tag{4}$$

or equivalently

$$\int_{\Omega}\int_{t}\frac{\partial T}{\partial t}d\Omega dt = \mu\int_{\Omega}\int_{t}\bigtriangledown\left(\bigtriangledown T\right)d\Omega dt \tag{5}$$

Using the Gauss divergence theorem:

$$\int_{\Omega}\int_{t}\frac{\partial T}{\partial t}d\Omega dt = \mu\int_{\partial\Omega}\int_{t}\left(\bigtriangledown T\right)d\partial\Omega dt \tag{6}$$

Considering T does not vary in the space, so a discretization to the left hand side of equation(6) is:

$$\int_{\Omega_i}\int_{t}\frac{\partial T}{\partial t}d\Omega_i dt \simeq P_i\frac{(T_i^{n+1}-T_i^n)}{\Delta t} \tag{7}$$

where P_i is the area of $i-th$ triangle.

Approximating the right hand size of (6) using the sides of the $i-th$ triangle:

$$\mu\int_{\partial\Omega}\int_{t}\left(\bigtriangledown T\right)d\partial\Omega dt \simeq \mu\left((T_{i1}^n-T_i^n)\frac{\lambda_{j1}}{\delta_{j1}}+(T_{i2}^n-T_i^n)\frac{\lambda_{j2}}{\delta_{j2}}+(T_{i3}^n-T_i^n)\frac{\lambda_{j3}}{\delta_{j3}}\right) \tag{8}$$

where λ_j is the size of $j-th$ side, δ_j is the distance between the centers (as defined in Section 2) of the triangle that share the $j-th$ side and the triangles i_p $(p=1,2,3)$ share the side j_p $(p=1,2,3)$ with the element i, how we can observe in Fig. 7.

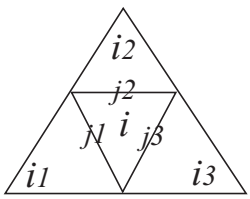


Fig. 7. Triangle and triangle side notations

Using (7) and (8), a implicit approximation to (6) can be written by:

$$P_i(T_i^{n+1}-T_i^n) = \mu\Delta t\left((T_{i1}^{n+1}-T_i^{n+1})\frac{\lambda_{j1}}{\delta_{j1}}+(T_{i2}^{n+1}-T_i^{n+1})\frac{\lambda_{j2}}{\delta_{j2}}+(T_{i3}^{n+1}-T_i^{n+1})\frac{\lambda_{j3}}{\delta_{j3}}\right) \tag{9}$$

Thus,

$$(T_i^{n+1} - T_i^n) = \frac{\mu \Delta t}{P_i} \left((T_{i1}^{n+1} - T_i^{n+1}) \frac{\lambda_{j1}}{\delta_{j1}} + (T_{i2}^{n+1} - T_i^{n+1}) \frac{\lambda_{j2}}{\delta_{j2}} + (T_{i3}^{n+1} - T_i^{n+1}) \frac{\lambda_{j3}}{\delta_{j3}} \right) \tag{10}$$

considering $\frac{\mu \Delta t}{P_i} = w_i$:

$$T_i^{n+1} - w_i (T_{i1}^{n+1} - T_i^{n+1}) \frac{\lambda_{j1}}{\delta_{j1}} - w_i (T_{i2}^{n+1} - T_i^{n+1}) \frac{\lambda_{j2}}{\delta_{j2}} - w_i (T_{i3}^{n+1} - T_i^{n+1}) \frac{\lambda_{j3}}{\delta_{j3}} = T_i^n \tag{11}$$

Isolating the T_i^{n+1} therm, the results above can be written in the matrix form as:

$$\left[1 + w_i \left(\frac{\lambda_{j1}}{\delta_{j1}} + \frac{\lambda_{j2}}{\delta_{j2}} + \frac{\lambda_{j3}}{\delta_{j3}} \right) \right] \cdot T_i^{n+1} - w_i \frac{\lambda_{j1}}{\delta_{j1}} T_{i1}^{n+1} - w_i \frac{\lambda_{j2}}{\delta_{j2}} T_{i2}^{n+1} - w_i \frac{\lambda_{j3}}{\delta_{j3}} T_{i3}^{n+1} = T_i^n \tag{12}$$

The matrix assembly is done using the mesh information of the entire domain. Each internal triangle of the domain generates one row of the coefficient matrix, where the number of terms depends on the number of neighbor cells of the triangle corresponding to that row of the matrix. The matrix generated is sparse, with a maximum of 4 non-null elements per row, and stored in CSR format. As the local matrices are sparse and non-symmetric, we used the MG-Schur with GMRES(m), with $m = 5$.

The solution of the heat transfer equation was obtained for a square domain, with $1m^2$ and $\mu = 0.1$. This domain was discretized in a 4 level mesh hierarchy generated by *MGTool*. The mesh levels contain 1337, 5348, 21392 and 85568 triangles, respectively. In this test we used constant boundaries conditions with 0^oC on three edges of the square and one edge with 1^oC.

6 Results

The implementations of this paper are being developed to explore parallelism in clusters. Conceptually, cluster is a collection of computers (workstations, personal computers or SMPs), called nodes, which are used exclusively for achieve high performance [5]. These machines are physically interconnected by a local network or a high performance network. The use of this architecture has a significant increase in the last years due, mainly, the low cost and the system scalability.

The implemented algorithms were run in the cluster of the Laboratory of Technology in Clusters (LabTeC) of UFRGS Computer Science Institute, developed in association with the Dell Computers. The cluster labtec is formed by 21 nodes, where 20 are dedicated exclusively for processing and one server node. The interconnection of the processing nodes is made through one Gigabit Ethernet switch. Each processing node is a Dual Pentium III 1.1 GHz, with 1 GB of RAM memory, 512 KB of cache and 18 GB SCSI hard disk; the server node is a Dual Pentium IV Xeon 1.8 GHz, with 1 GB of memory RAM and 36 GB SCSI hard disk.

In this type of architecture parallel programming is usually explicit, requiring complete control over implementation strategies and over the implementation itself, and, in this context, parallelism is obtained through the division-conquer paradigm. From the programming point of view, the SPMD (Single Program Multiple Data) paradigm was used. For the development of parallel applications in distributed memory machines, like clusters, is necessary the use of an message passage library. All the algorithms were implemented using C programming language and the MPICH 1.2.5 message passing library [11] over Linux operating system.

The MG-Schur method was tested up to 38 processes. We also compare the MG-Schur to a Schur complement method without multigrid. The Figs. 8 and 9 show execution time and computational efficiency obtained, respectively. These results were obtained without considering the opening and reading of input data and considering 15 cycles, where each cycle is composed by the maximum number of iterations necessary to reach the desired accuracy. In the tests the error is 10^{-5}.

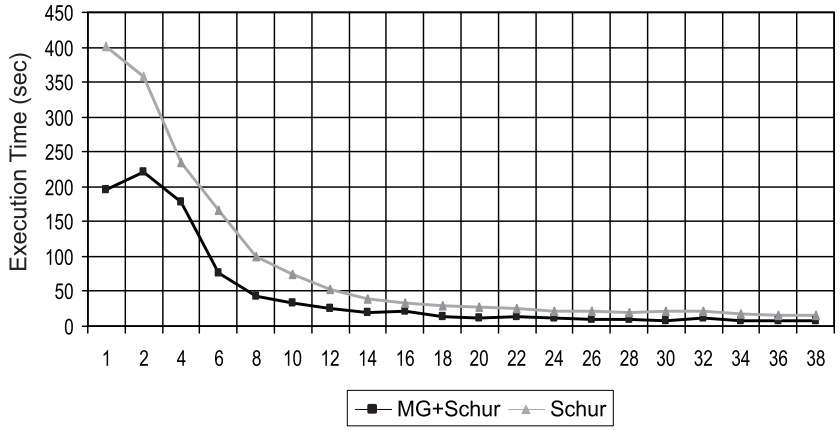


Fig. 8. Execution time x number of processes

Accordly to the figures, we can observe that both tested methods are scalable but the MG-Schur is almost twice faster than Schur method. The maximum efficiency was achieved when executed with 20 processes, where we obtained 16.11 of speedup and 79% of efficiency. The maximum speedup achieved was 27.07 using 36 processes, with 75% of efficiency.

It should be noted, that the efficiency decreases when using some amount of processes (e.g 2, 4, 16, 28 and 32). This fact occurs due to the solution of the large Schur complement interface systems, that are generated when the domain partitioning presents large artificial boundaries among the subdomains.

As the simulation runs, heat is transmitted from warmer parts of the domain to cooler parts of the domain, and gradually converging to a stable state. The presented solution is continuous among the subdomains, showing a effective par-

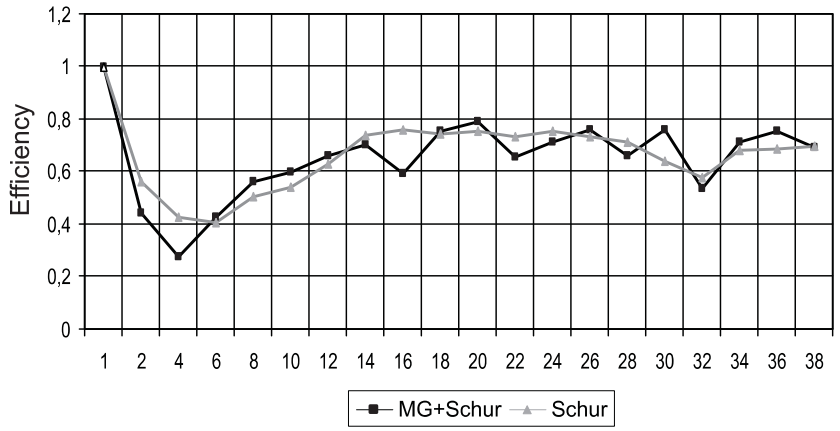


Fig. 9. Efficiency x number of processes

allel solution. In the Fig. 10 we can observe the heat distribution in a section of the domain (x=0.5).

7 Conclusion and Future Work

In this work we presented a parallel implementation of a multigrid method, using Schur complement domain decomposition method. The numerical results obtained when running the solver in several processors are consistent with the results obtained running it in one processor, and the difference between the results are due to the accuracy desired.

The experiments performed have shown that the proposed implementation has shown to be computationally efficient, good scalability, and good numerical quality.

As future work, three important issues will subject of more research: implementation of a flexible multigrid algorithm to allow the solution by diverse types of multigrid cycles, parallelization of the multigrid by overlapping domain decomposition, and adapt *MG-Tool* to the solution of the hydrodynamics of UnHidra model. The UnHidra is a multi-physics parallel computational model for the simulation of substance transport and for the 2D and 3D hydrodynamic flow in water bodies, using unstructured meshes.

8 Acknowledgments

Thanks to CAPES for financed the master thesis related to this work. Some issues of this research are included to the UnHIDRA project (CT-HIDRO, Processo 502858/2003-6, Edital MCT/CNPq/CT-HIDRO 01/2003).

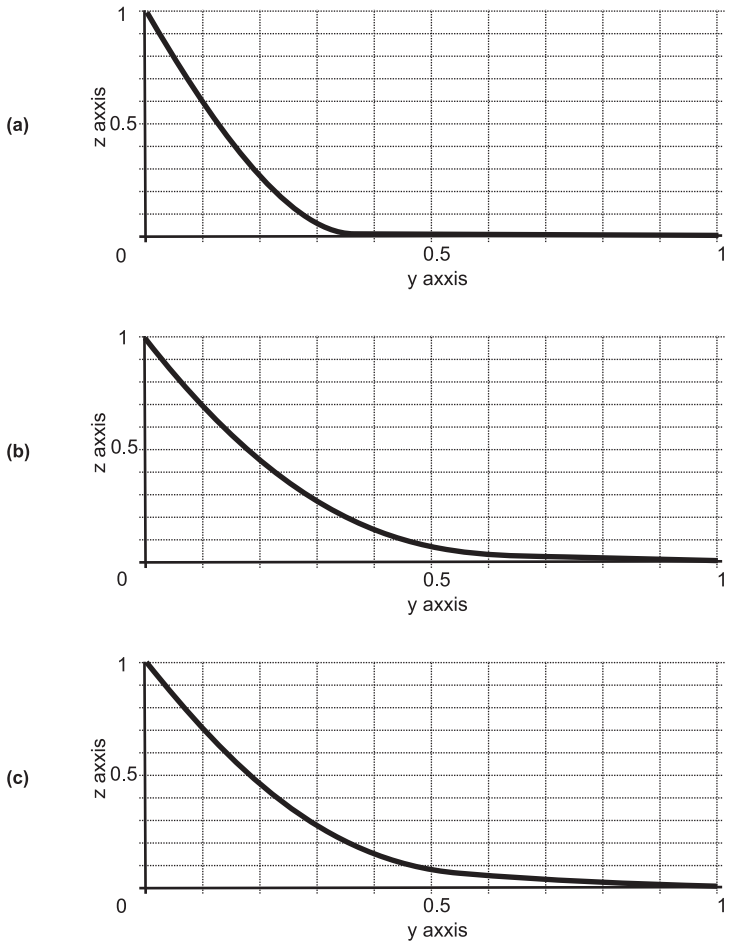


Fig. 10. Three steps in the heat transfer problem. From top to bottom, $t = 1$, $t = 5$ and $t = 10$. We can observe that the system gradually converge to a stable state

References

1. S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.
2. T. Barth. Numerical methods and error estimation for conservation laws on structured and unstructured meshes. Von Karman Institute, LS04-2003, 2003, pp 1-65, 2003.
3. P. Bastian, W. Hackbusch, and G. Wittum. Additive and multiplicative multi-grid: a comparison. Computing, 60:345364, 1998.
4. W. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, 1987.
5. R. Buyya. *High Performance Cluster Computing: Architecture and Systems*, volume 1. Prentice Hall, 1999.

6. V. Casulli and R. Walters A. An Unstructured Grid, Three-Dimensional Model based on the Shallow Water Equations. International journal for numerical methods in fluids, v. 3, p. 331-348. 2000.
7. T. F. Chan and B. F. Smith. Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes. Electronic Transactions on Numerical Analysis. Volume 2, pp. 171-182, December 1994.
8. E. Chow, R. Falgout, J. Hu, R. Tuminaro and U. Meier Yang. A Survey of Parallelization Techniques for Multigrid Solvers to appear in Frontiers of Parallel Processing For Scientific Computing, SIAM book series , 2005.
9. C. C. Douglas. MGNet: a multigrid and domain decomposition network. *http://www.mgnet.org*.
10. M. Filipiak. Mesh Generation. [S.l.]: EPCC, Edinburgh, 1996. Watch Report.
11. W. Groop, et al. A High Performance, Portable Implementation of the MPI Message Passing Interface Standard. Parallel Computer, v.22, n.6, p.789-828, Sep. 1996.
12. G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
13. B. Niceno. EasyMesh(Version 1.4), freely available mesh generator on the web site: http://www-dinma. univ.trieste.it/∼nirftc/research/easymesh/.
14. P.-O. Fjällström. Algorithms for Graph Partitioning: a survey. In *Linköping Electronic Articles in Computer and Information Science*, volume 3, Linköping, 1998. Department of Computer and Information Science, Linköping University.
15. Y. Saad and M. Sosonkina Distributed Schur Complement Techniques for General Sparse Linear Systems. SIAM Journal on Scientific Computing. October, 1997.
16. Y. Saad and M. Sosonkina. pARMS: a package for solving general sparse linear systems of equations. In R. Wyrzykowski, J. Dongarra, M. Paprzycki, and J. Wasniewski, editors, *Parallel Processing and Applied Mathematics*, volume 2328 of *Lecture Notes in Computer Science*, pages 446–457, Berlin, 2002. Springer-Verlag.
17. B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Pres, Cambridge, 1996.
18. U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid.* 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.
19. P. Wesseling. *Introduction to Multigrid Methods.* John Wiley & Sons, Chichester, 1992.