Performance Comparison of Parallel Geometric and Algebraic Multigrid Preconditioners for the Bidomain Equations

Fernando Otaviano Campos, Rafael Sachetto Oliveira, and Rodrigo Weber dos Santos

FISIOCOMP: Laboratory of Computational Physiology, Department of Computer Science - Universidade Federal de Juiz de Fora (UFJF), PO Box 15.064-91.501-970 - Juiz de Fora - MG - Brazil
fernando.ocampos@terra.com.br, rsachetto@gmail.com, rodrigo@dcc.ufjf.br

Abstract. The purpose of this paper is to discuss parallel preconditioning techniques to solve the elliptic portion (since it dominates computation) of the bidomain model, a non-linear system of partial differential equations that is widely used for describing electrical activity in the heart. Specifically, we assessed the performance of parallel multigrid preconditioners for a conjugate gradient solver. We compared two different approaches: the Geometric and Algebraic Multigrid Methods. The implementation is based on the PETSc library and we reported results for a 6-node Athlon 64 cluster. The results suggest that the algebraic multigrid preconditioner performs better than the geometric multigrid method for the cardiac bidomain equations.

1 Introduction

The bidomain formulation [1] is currently the model that best reflects the electrical activity in the heart. The non-linear system of partial differential equations (PDEs) models both the intracellular and extracellular domains of cardiac tissue from an electrostatic point of view. The coupling of the two domains is done via non-linear models describing the current flow through the cell membrane.

Unfortunately, the bidomain equations are computationally very expensive. Efficient ways of solving the large linear algebraic system that arises from the discretization of the bidomain model have been a topic of research since 1994 [2]. Many different approaches among direct and iterative methods have been employed considering the problem's size and the computing resources available. However, iterative methods such as conjugate gradient (CG) are generally more scalable.

In previous works we have compared different parallel preconditioner methods for the Conjugate Gradient iterative algorithm. In [3] we have shown that preconditioners based on the Geometric Multigrid Method (GMG) performed better than the classical incomplete LU (ILU) preconditioners for 2D and 3D cardiac

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part I, LNCS 3991, pp. 76–83, 2006.

electric propagation problems. In [4] the GMG preconditioners were compared to different Additive Schwarz (ASM) preconditioners. The results taken from a 16-node HP-Unix cluster indicated that the multigrid preconditioner was at least 13 times faster than the single-level Schwarz based techniques and requires at least 11% less memory.

In this paper we compare our previous parallel implementation of the GMG preconditioner [3], [4] to an Algebraic Multigrid (AMG) based parallel preconditioner [5]. We focus on the solution of the linear algebraic system associated with the elliptic part of the bidomain model, since this part dominates computation. We employ the CG method preconditioned with both, Geometric (GMG) and Algebraic (AMG) parallel multigrid (MG) techniques. The implementation is based on the PETSc C library [6] (which uses MPI) and is tested on problems involving thousands of unknowns. The results taken from a 6-node Athlon 64 Linux cluster indicate that the AMG preconditioner is at least 3 times faster than GMG and 117 times faster than the traditional ILU preconditioner.

2 Mathematical Formulation

The set of Bidomain equations[1] is currently one of the most complete mathematical models to simulate the electrical activity in cardiac tissue:

$$\chi\left(C_m\frac{\partial\phi}{\partial t}+f(\phi,t)\right)=\nabla.(\sigma_i\nabla\phi)+(\sigma_e\nabla\phi_e), \tag{1}$$

$$\nabla.((\sigma_e+\sigma_i)\nabla\phi_e)=-\nabla.(\sigma_i\nabla\phi), \tag{2}$$

$$\frac{\partial v}{\partial t}=g(\phi,\boldsymbol{\eta}),\quad \phi=\phi_i-\phi_e. \tag{3}$$

Where ϕ_e is the extracellular potential, ϕ_i the intracellular potential and ϕ is the transmembrane potential. Eq. (3) is a system of non-linear equations that accounts for the dynamics of several ionic species and channels (proteins that cross cell membrane) and their relation to the transmembrane potential. The system of (3) typically accounts for over 20 variables, such as ionic concentrations, protein channel resistivities and other cellular features. σ_i and σ_e are the intracellular and extracellular conductivity tensors, i.e. 3×3 symmetric matrices that vary in space and describe the anisotropy of the cardiac tissue. C_m and χ are the cell membrane capacitance and the surface-to-volume ratio, respectively.

Unfortunately, a solution of this large nonlinear system of partial differential equations (PDEs) is computationally expensive. One way to solve (1)-(3) at every time step is via the operator splitting technique [4]. The numerical solution reduces to a three step scheme which involves the solutions of a parabolic PDE, an elliptic PDE and a nonlinear system of ordinary differential equations (ODEs) at each time step. Rewriting equations (1)-(3) using the operator splitting technique (see [4] for more details) we get the following numerical scheme:

$$\varphi^{k+1/2}=(1+\Delta tA_i)\varphi^k+\Delta tA_i(\varphi_e)^k; \tag{4}$$

$$\varphi^{k+1}=\varphi^{k+1/2}-\Delta tf(\varphi^{k+1/2},\boldsymbol{\zeta^k})/(C_m), \tag{5}$$

$$\boldsymbol{\zeta^{k+1}} = \boldsymbol{\zeta^{k}} + \Delta tg(\varphi^{k+1/2}, \boldsymbol{\zeta^{k}});$$
$$(A_i + A_e)(\varphi_e)^{k+1} = -A_i\varphi^{k+1}. \quad (6)$$

Where φ^k, φ_e^k and $\boldsymbol{\zeta^k}$ discretizes ϕ, ϕ_e and $\boldsymbol{\eta}$ at time $k\ \Delta_t$; A_i and A_e are the discretizations for $\nabla.((\sigma_i\nabla)/(\chi C_m)$ and $\nabla.((\sigma_e\nabla)/(\chi C_m)$, respectively. Spatial discretization was done via the Finite Element Method using a uniform mesh of squares and bilinear polynomials as previously described in [4].

Steps (4), (5) and (6) are solved as independent systems. Nevertheless, (4), (5) and (6) are still computationally expensive. One way of reducing the time spent on solving these equations is via parallel computing.

3 Parallel Multigrid Preconditioners

In the previous Section we presented a mathematical model for the electrical potential in the cardiac tissue. Many direct and iterative approaches have been employed in the solution of the linear systems that appear after the spatial discretization of the Bidomain equations. Direct factorization methods such as Cholesky or LU performed better than iterative methods for small simulation setups [7], i.e., when memory limitations were not a concern. However, for larger problems, for instance, the simulation of a whole three-dimensional (3D) heart in which the discretization leads to millions of nodes, an iterative method is mandatory. The preconditioned CG method has become the standard choice for an iterative solver of the bidomain formulation.

We used CG to solve the linear system and intended to compare both GMG and AMG as preconditioners. The solution of (4)-(6) was implemented in parallel using the PETSc C library, which uses MPI. CG is parallelized via linear domain decomposition. The spatial domain is decomposed into *proc* domains with equal sizes, where *proc* is the number of processors involved in the simulation. In addition, we compared the traditional block incomplete LU parallel preconditioner (ILU) against the multigrid ones.

The basic idea behind multigrid methods relies on the fact that simple iterative methods are very efficient at reducing high-frequency components of the residual, but are very inefficient with respect to the lower frequency components. In the classical MG theory [8], such iterative methods are called smoothers, since they smooth the error better than reduce its average. The multilevel solution to this problem is to project the residual onto a smaller space, a coarser grid of the problem, where the lower spatial frequency components can be handled more efficiently. The problem is now solved on the coarser grid and the residual is then projected back to the original space, the finer grid. This way, the residual on the finer grid has an approximation of the lower frequency components removed, and the convergence of the iterative method is faster.

In our GMG preconditioner implementation we successively generated coarser regular grids based on the finest regular grid G_0, i.e. the uniform element mesh. This procedure was repeated until the coarsest level, $G_{levels-1}$. For each grid pair, G_l and G_{l+1}, a prolongation rectangular matrix, P_l, was generated using

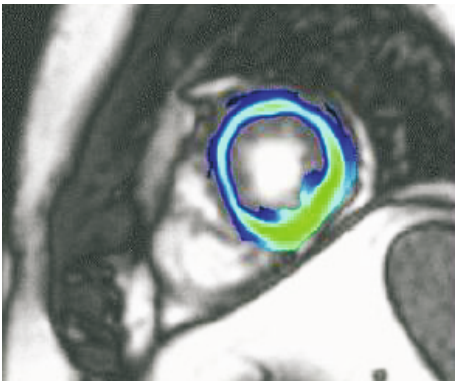

Fig. 1. Simulated electrical wave propagation overlapped to the Resonance Image

a bilinear interpolation scheme. For every grid level ($l = 0$ to $l = levels - 1$), a matrix A_l was generated by applying the finite element method to the particular grid. Further details of our GMG implementation can be found in [3].

The mainly difference between GMG and AMG preconditioners is the coarse grid selection. In the GMG scheme, just simple slices are made to the fine grid creating a coarse grid with half of the nodes in each direction. To select the coarse grid points in the AMG, we seek those unknowns u_i which can be used to represent the values of nearby unknowns u_j. It is done via the concepts of dependence and influence. We say that the point i depends on the point j or j influences i, if the value of the unknown u_j is important in determining the value of u_i from the ith equation. Based on measures of dependence and influence taken from the matrix coefficients, special heuristics generate the maximal subset of points of the coarse grid. Thus, the process of coarse matrices generation depends solely on the original finest-grid matrix. Different from the GMG, the finite element method is only used once in the AMG method, i.e. during the creation of the finest-grid matrix. All the other matrices are obtained algebraically.

In this work we adopted the parallel AMG code BoomerAMG [5] with its Falgout-coarsening strategy.

In both GMG and AMG preconditioners the smoother used for all but the coarsest level was an iterative method. For the coarsest level, we used a direct solver. This was not done in parallel, i.e., it was repeated on every processor, avoiding any communication.

4 Results

We performed several tests in order to compare the different preconditioners on a 6 node Linux Cluster, each node equipped with a AMD Athlon 64 3 GHz processor, 2 GB of RAM and connected by 1 $Gbit/s$ Ethernet switch. In all tests, we simulated the cardiac electric propagation on a two-dimensional cut of the left ventricle obtained during the cardiac diastole phase by the resonance magnetic

technique of a healthful person. After segmenting the resonance image, a two-dimensional mesh of 769×769 points was generated, that models the cardiac tissue, blood and torso.

All bidomain parameters were taken from [9]. The capacitance per unit area and the surface area - to- volume ratio are set to $1\,mF/cm^2$ and $2000/cm$, respectively. The interface between cardiac tissue and bath is modeled as described in [10]. All the other boundaries are assumed to be electrically isolated. The spatial and temporal discretization steps of the numerical model were set to $0.0148\,cm$ and $0.05\,ms$, respectively. The simulation was carried out for $5\,ms$, or 100 time steps, after a single current stimulus was introduced at a selected endocardial site. For simulating the action potential of cardiac cells we used the human ventricular model of ten Tusscher et al. [11].

The stop criterion adopted for all the preconditioned CG algorithms was based on the unpreconditioned and absolute L2 residual norm, $\|Ax_i - b\|^2 \leq tol$, where x_i was the solution at iteration i and tol was a tolerance which was set to 10^{-6}. Although this is not the most efficient stop criteria for the CG, it is the fairest one when comparing different preconditioning methods.

The performance measurements reported in this section, such as the execution time, CG number of iterations, average time per iteration, setup time and number of nonzeros in an particular grid are related to the solution of the elliptic part (2), since this part is responsible for around 80% of the whole simulation time. The memory usage is related to the whole model.

4.1 Parameter Tuning

We tuned the following parameters: $fill$ for ILU; $levels$ for GMG; $levels$ and $strongthreshold$ for AMG. Table 1 shows, for different numbers of processors, the optimal parameter values that yielded the fastest execution time. The parameter $fill$ was varied from 0 to 4; GMG $levels$ varied from 2 to 6; AMG $levels$ from 6 to 16 and $strongthreshold$ was set to 0.25, 0.50 and 0.75. In addition, all parameters were tuned for best execution time on 1, 2, 4 and 6 processors. A total of 160 simulations were performed during about one week of computation time.

Due to the long execution time demanded for the ILU preconditioner, just simulations on 6 processors were performed for this case. With this number of processors the optimal value of $fill$ was 4.

For GMG, the optimal value of $levels$ depended on $proc$. On a single processor, $levels = 2$ corresponded to the fastest execution. In parallel, however, since the coarsest grid is solved sequentially, the cost of fewer grid $levels$ rivaled the gains of parallelism. Therefore, as $proc$ increased, the optimal $levels$ also increased to 3.

The AMG preconditioner performed better with the $strongthreshold$ set to 0.25, i.e. the smallest experimented value. The choice of the $strongthreshold$ value directly influences the number of grid points in each level, i.e. the number of non-zero elements of each matrix A_l. High $strongthresholds$ generated rich coarse matrices in terms of the information that is kept from the finest level. This contributed towards faster convergence in terms of iteration count. However,

Table 1. Number of nonzeros on 1 and 6 processors

proc	levels	GMG	AMG
1	2	1329409	1722609
	8	-	8138
6	3	332929	683411
	8	-	8707

Table 2. Comparison between GMG and AMG on 1 and 6 processors

Type	proc	Time (s)	CG Iters.	Time / Iter
ILU		-	-	-
GMG	1	1867.05	1050	1.78
AMG		626.27	916	0.68
ILU		17502.88	175481	0.10
GMG	6	578.41	1290	0.45
AMG		141.12	900	0.16

this improvement did not result in faster execution times, since every level was considerably more computationally expensive.

AMG performed better with more *levels* ($levels = 8$ was the fastest) than GMG. Many factors may have contributed to this. It was shown before that AMG coarsening algorithms tend to coarsen in the direction of the dependence and perform better than the traditional geometric algorithms when the problem has anisotropic or discontinuous coefficients. Cardiac tissue is highly anisotropic and the conduction coefficients of the bidomain model reflect the cardiac tissue properties. Therefore, fewer levels in the GMG may be necessary in order to avoid loss of anisotropic information. In addition, the implementation of the AMG and GMG preconditioners differs in some aspects. The *smoothers* (the iterative methods used) are different. GMG uses a more robust and expensive method, a preconditioned CG, to *smooth* the residuals in all but the coarsest level. AMG uses a simple relaxation method. Thus, every GMG level is more computationally expensive than an AMG one. The direct methods used to solve the coarsest level are also different. GMG uses a more efficient and fast direct LU solver with nested dissection reordering. The AMG direct solver was very inefficient in handling large problems, i.e. coarsest matrices with less than 5 *levels*.

In summary, compare to GMG, AMG performed better with more levels and fewer non-zero elements in the coarsest matrix. Table 1 shows the number of non-zeros of the coarsest matrix of the simulations with 1 and 6 processors, for the AMG and GMG cases.

4.2 Performance Comparison and Parallel Speedup

Table 2 presents the execution time, total number of CG iterations and average time per iteration for all time steps (100 time steps) per processor for all preconditioners with the optimal parameters. Both GMG and AMG preconditioners

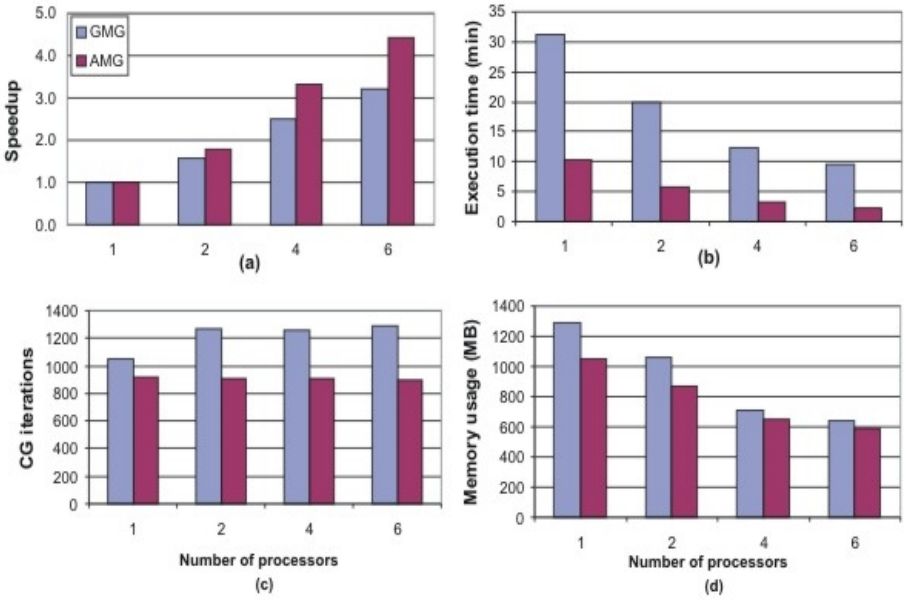

Fig. 2. Computational requirements for 2D anisotropic systole phase. (a) parallel speedup relative to single processor execution time for GMG and AMG preconditioners for different number of processors. (b) execution time, (c) total number of CG Iterations for 100 time steps and (d) memory usage.

achieved better performance results than ILU on 6 processors. GMG was 30.26 times faster than ILU. In the same conditions, AMG was 124.03 times faster than ILU. In addition, ILU took more than 17 thousands CG iterations to solve the problem, i.e. for the total 100 time steps. Although ILU presented a better average time/iteration than the multigrid methods, it needed much more CG iterations to solve the system. GMG and AMG converged with around one thousand iterations, i.e. 10 iterations per time step.

AMG was 2.98 (4.10) times faster than GMG on 1 (6) processors, and required 50% (54%) less on setup phase. When using the GMG preconditioner, the number of CG iterations increased as the number of processors scaled up from 1 to 6. The AMG method was more stable and the number of iterations did not increase when increasing the number of processors. Figure 2 shows the comparison between GMG and AMG preconditioners with 1, 2, 4 and 6 processors. Both multigrid preconditioners achieved reasonable parallel speedup (execution time on 1 processor/execution time) results, 3.23 for GMG and 4.44 for AMG on 6 processors. AMG performed better for all tests, it converged faster (took less CG iterations), required around 22% less memory with $proc \leq 2$ and 9% when $proc \geq 4$. Finally, according to the speedup results, AMG achieved a better scalability on 6 processors, which resulted from the smaller coarsest grid matrices. The direct method used in the coarsest grid is not parallelized. Thus, the larger matrices used by GMG in this level (see table 1) severely degrade the parallel speedup.

4.3 Conclusions

In this work, we employed the conjugate gradient algorithm for the solution of the linear system associated with the elliptic part of the bidomain equations and compared two different parallel multigrid preconditioners: the GMG and AMG. The results taken from a 6-node Athlon 64 Linux cluster indicate that the algebraic multigrid preconditioner is at least 3 times faster than GMG, requires at least 9% less memory and achieved a better scalability on 6 processors.

Acknowledgements. We thank the support provided by the Universidade Federal de Juiz de Fora (UFJF-Enxoval) and the Brazilian Ministry of Science and Technology, CNPq (processes 506795/2004-7 and 473872/2004-7).

References

1. Sepulveda N. G., Roth B. J., and Wikswo Jr. J. P.: Current injection into a two-dimensional anistropic bidomain. Biophysical J. **55** (1989) 987–999
2. Hooke N., Henriquez C., Lanzkron P., and Rose D.: Linear algebraic transformations of the bidomain equations: Implications for numerical methods. Math. Biosci. vol. 120, no. 2, pp. (1994) 127–45
3. Weber dos Santos R., PLANK G., BAUER S., and VIGMOND E. J.: Parallel Multigrid Preconditioner for the Cardiac Bidomain Model IEEE Trans. Biomed. Eng. **51(11)** (2004) 19601968
4. Weber dos Santos R., Plank G., Bauer S., and Vigmond E.: Preconditioning techniques for the bidomain equations. Lecture Notes In Computational Science And Engineering. **40** (2004) 571–580
5. Henson V. E., and Yang U. M.: BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner. Technical Report UCRL-JC-139098, Lawrence Livermore National Laboratory, (2000)
6. Balay S., Buschelman K., Gropp W., Kaushik D., Knepley M., McInnes L., Smith B., and Zhang H.: PETSc users manual. Technical report ANL-95/11 - Revision 2.1.15, Argony National Laboratory, (2002)
7. Vigmond E., Aguel F., and Trayanova N.: Computational techniques for solving the bidomain equations in three dimensions Trans. Biomed. Eng. IEEE. **49** (2002) 1260–9
8. Briggs W., Henson V., and McCormick S.: A Multigrid Tutorial. SIAM, Philadelphia, PA, Tech. Rep., (2000).
9. Jones, J. Vassilevski, P.: A parallel graph coloring heuristic. SIAM J. Sci. Comput. **14** (1993) 654–669
10. Krassowska W., and Neu. J. C.: Effective boundary conditions for syncytial tissues. IEEE Trans. Biomed. Eng **41** (1994) 143–150.
11. ten Tusscher K. H. W. J., Noble D., Noble P. J., and Panfilov A. V.: A model for human ventricular tissue J. Physiol. **286** (2004) 1573–1589