Three-Dimensional Cardiac Electrical Activity Simulation on Cluster and Grid Platforms *

J. M. Alonso[1], J. M. Ferrero (Jr.)[2], V. Hernández[1], G. Moltó[1], M. Monserrat[2], and J. Saiz[2]

[1] Departamento de Sistemas Informáticos y Computación.
Universidad Politécnica de Valencia. Camino de Vera s/n 46022 Valencia, Spain
{jmalonso,vhernand,gmolto}@dsic.upv.es
Tel. +34963877356, Fax +34963877359
[2] Departamento de Ingeniería Electrónica.
Universidad Politécnica de Valencia. Camino de Vera s/n 46022 Valencia, Spain
{cferrero,monserr,jsaiz}@eln.upv.es
Tel. +34963877600, Fax +34963877609

Abstract. Simulation of action potential propagation on cardiac tissues represents both a computational and memory intensive task. The use of detailed ionic cellular models, combined with its application into three-dimensional geometries turn simulation into a problem only affordable, in reasonable time, with High Performance Computing techniques. This paper presents a complete and efficient parallel system for the simulation of the action potential propagation on a three-dimensional parallelepiped modelization of a ventricular cardiac tissue. This simulator has been integrated into a Grid Computing system, what allows an increase of productivity in cardiac case studies by performing multiple concurrent parallel executions on distributed computational resources of a Grid.

1 Introduction

Cardiac arrhythmias are one of the first causes of mortality in developed countries. Among them, ventricular tachycardias and ventricular fibrillation stand out because of triggering sudden cardiac death. In spite of intense research, the mechanisms of generation, maintenance and termination of these arrhythmias are not clearly understood.

Recently, mathematical models of the propagation of cardiac electrical activity are being considered as a powerful and helpful tool to better understand the mechanisms involved in the development of ventricular arrhythmias. The electrical activity of cardiac cells is described by detailed models of ion movements through the cell membrane. By combining the mathematical formulation

* The authors wish to thank the financial support received from (1) the Spanish Ministry of Science and Technology to develop the projects CAMAEC (TIC2001-2686) and GRID-IT (TIC2003-01318), and (2) the Universidad Politécnica de Valencia for the CAMAV project (20020418).

of membrane ion kinetics and the structural complexity of the heart it is possible to simulate in computers the complex electrical propagation in cardiac tissues.

Earlier studies characterised this tissue as a simple one dimensional fiber. A more realistic model consisted of a thin sheet of myocardium, where one cell was connected to its four neighbours in a two-dimensional regular structure. However, to study the complex dynamics underlying the formation and development of ventricular tachycardias and ventricular fibrillation, a 3D model of a cardiac tissue with appropriate dimensions is required. This 3D tissue consists of a very large number of cardiac cells (typically hundreds of thousands), governed by time-consuming ionic models which require tenths of state variables for each cell. Besides, provided that the simulation periods are typically milliseconds and even seconds, integrated with time steps of a few μs, an action potential propagation simulation can last for several days or even weeks on a traditional serial platform.

In addition to all the computational requirements for a single simulation, there are many cardiac case studies that demand the execution of several simulations. For example, testing the effects of new drugs requires the execution of multiple parametric simulations, where for instance the drug concentration is changed. As another example, studying the effects of late ischemia, it is necessary to vary the junctional resistances in a determined interval and observe the evolution of the electrical activity of the tissue under different anisotropy conditions.

Therefore, to harness all this computational burden, we have integrated two different technologies. First of all, High Performance Computing techniques offer the possibility to reduce the execution time of a single simulation, as well as to enable the simulation of larger three-dimensional tissues during longer time by performing execution on a cluster of PCs. On the other hand, Grid Computing technology emerges as a solution for the collaborative usage of multi-organisational computational resources [1]. In this work, both techniques have been integrated into a system that allows concurrent parallel simulations of action potential propagation on remote, multi-organisational resources of a Grid infrastructure, based upon the public domain Globus Toolkit [2] and the commercial InnerGrid [3] middlewares.

The article is structured as follows. Section 2 presents the underlying mathematical model. Then, section 3 explains the parallelisation approach implemented in order to reduce the execution time of a single simulation. Next, section 4 discusses the performance results achieved. Later, section 5 describes the Grid Computing approaches with both middlewares. Section 6 presents a case study to analyse the Grid advantages and finally section 7 summarizes the main achievements.

2 Mathematical Model and Geometry

The action potential propagation on a monodomain modelization of a cardiac tissue can be described by the following partial derivative equation:

$$\nabla \cdot \sigma \nabla V_m = C_m \cdot \frac{\partial V_m}{\partial t} + I_{ion} + I_{st} \quad (1)$$

where σ represents the conductivity tensor, V_m is the membrane potential of the cells, C_m stands for the membrane capacitance, I_{st} represents an stimulus current to provoke an action potential and I_{ion} is the sum of ionic currents traversing the membrane of each cell, computed by the comprehensive and detailed Luo-Rudy Phase II ionic model [4].

The term action potential denotes a transient change of the membrane potential caused by the electrically excitable heart cells. When a stimulus applied to the cell leads to depolarization of resting membrane potential up to its threshold, then an action potential is induced. This response is characterised by an initially fast rise of the membrane potential followed by a slow recovery to the resting potential.

In our three-dimensional modelization, the ventricular tissue cardiac cells are linked with resistances within a parallelepiped geometry. Cardiac muscle fibers are assumed to have faster longitudinal than transversal or transmural conductivity, accounting for the anisotropy condition of a ventricular cardiac tissue.

Equation (1) is spatially discretized using a seven-point finite difference stencil and employing the Crank-Nicholson's semi-implicit method, what leads to the following algebraic equation:

$$G_L \cdot V_m^{t+1} = G_R \cdot V_m^t + I_{ion}^t + I_{st}, \forall t = 1, 2, ..., n \ . \quad (2)$$

The matrices G_L and G_R account for the conductivity along the cells of the tissue. The I_{ion} term encapsulates the cellular ionic model, requiring the resolution of several time-dependent ordinary differential equations. Thus, the simulation turns into an iterative process where the membrane potential of the cells is reconstructed through the resolution of a large sparse linear equation system for each simulation time step.

Even though there have been several parallel approaches to this computational problem [5], the good efficiency results achieved on a beowulf cluster, logically based on a distributed memory paradigm, together with appearing to be the first simulation system to approach both a parallel and a Grid Computing philosophy represent a step forward in the study of the electrical activity of the heart.

3 Parallel Solution

3.1 Parallelization Approach

The cells in our three-dimensional parallelepiped are numbered following a natural ordering and assigned to the processors by groups with contiguous numeration indexes of approximately the same size. This way, each processor is in charge of performing all the calculations corresponding to its part of the tissue.

3.2 Conductivity Matrix Generation

The conductivity matrices G_R and G_L are generated in parallel with no communication among the processors. These matrices, together with the ionic (I_{ion}), the stimulus (I_{st}) and membrane potential (V_m) vectors, have been partitioned among the processors following a rowwise block-striped distribution, what overcomes the memory constraints that may arise when simulating a large three-dimensional tissue on a single computer, thus enabling the simulation of larger tissues.

3.3 Cell Membrane Potential Calculation

In order to obtain the membrane potential of the cells, a large sparse linear equation system must be solved for each simulation time step. The G_L coefficient matrix is symmetric and positive definite, with a size equal to the number of cells in the tissue. For a 3D tissue of 1 million cardiac cells (100x100x100 cells), the coefficient matrix has dimension 1 million with 7 million nonzero elements.

Based upon the framework that the PETSc library [6] offers, two different strategies have been employed in order to solve this large sparse linear equation system. A parallel direct method, based on a Multifrontal Cholesky Factorization provided by the MUMPS library [7], integrated within PETSc, and a parallel iterative method, based on the Preconditioned Conjugate Gradient, have been tested. For the direct method, a previous step of ordering to reduce the fill-in is performed with the Multilevel Nested Dissection algorithm implemented in the METIS library [8].

3.4 Right-Hand Side Vector Generation

The right-hand side vector generation of the linear equation system has been fully parallelised. First of all, a distributed updating of the state of the cells and the computation of the I_{ion}^t term, via the Luo-Rudy Phase II model, take place. This represents the most time-consuming step of the simulation, where each processor only updates its local part of the tissue without any inter-process communication.

Then, a sparse matrix-vector product $G_R \cdot V_m^t$, with the data distribution described in section 3.2, must be carried out in parallel. Communications are needed in this stage taking into account the sparsity pattern of the G_R matrix, where each processor can demand the membrane potential of cells residing in its neighbour processors.

Next, the I_{st} stimulus vector is computed with no communications. Finally, the right-hand side vector is generated as a sum of three vectors, with no inherent communication cost.

4 Experimental Results of the Parallel Implementation

The simulations have been run on a cluster of 20 dual-processor 2 GHz Pentium Xeon with 1 GByte of RAM, interconnected with a 4x5 torus SCI network.

Figure 1 shows the execution time of a single time step, comparing the direct and the iterative method, when simulating an action potential propagation on a 50x50x50 cells tissue.

Regarding the direct method, the ordering, symbolic and numerical factorizations cost have been neglected because the coefficient matrix remains constant through the simulation and thus, they can be reused. Therefore, only the solution of the triangular systems is included in the cost of the direct method, as the time of the three steps would vanish in a long simulation. It should be taken into account that, for very large tissues, the factorization could exceed the available memory, what represents a handicap which an iterative method does not suffer from.

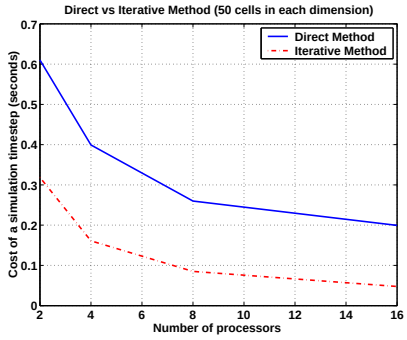


Fig. 1. Conjugate Gradient Method versus solution of the triangular systems after a Multifrontal Cholesky Factorization

The conjugate gradient method with no preconditioning has probed to be the best iterative solver tested. Besides, as Fig. 1 reflects, it performs twice as fast as the resolution of the triangular systems. In fact, the coefficient matrix is well conditioned and convergence is obtained within few iterations with a good residual tolerance.

Figure 2 shows the speedup and efficiency of the whole simulation system when simulating an action potential propagation on a 100x100x100 cells cardiac tissue during 250 ms, using the conjugate gradient method with no preconditioning, and employing up to 32 processors. Simulations have been performed running two processes per node. It should be pointed out that the simulation system scales quite linear with the number of processors.

For such a simulation, the execution times are reflected in Table 1. When using 32 processors, we have reduced a simulation that on a sequential platform would last more than two days to a couple of hours. Time results for one processor are not provided due to memory requirements.

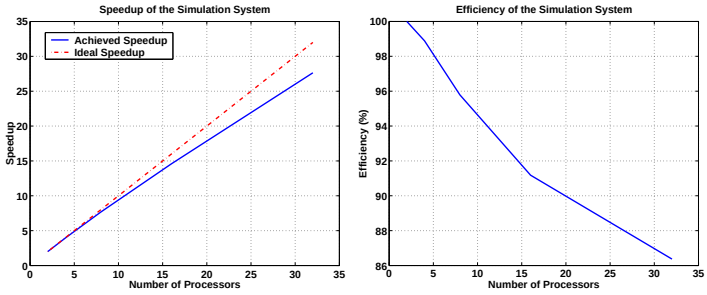


Fig. 2. Scalability of the simulation system

Table 1. Execution times and scalability results for a simulation of action potential propagation on a 100x100x100 cells tissue during 250 ms (dt = 8 μs), using up to 32 processors

Number of processors	Simulation time (hours)	Speedup	Efficiency
2	34.72	-	-
4	17.55	3.95	98.88
8	9.06	7.66	95.79
16	4.75	14.58	91.18
32	2.51	27.63	86.36

5 Grid Computing Approaches

In order to harness all the computational requirements that cardiac case studies require, which may overwhelm the resources of a single organisation, we have deployed two middlewares, a commercial solution provided by InnerGrid [3] software and the wide adopted public domain industrial standard Globus Toolkit [2].

Innergrid is a multi-platform software that enables the collective sharing of computational resources, within an organisation, in order to enlarge the productivity of parametric sequential jobs. This software offers a fault-tolerance scheme that guarantees the execution of the tasks as long as there are living nodes in the Grid. InnerGrid exposes a web interface from which the configuration and the management of the tasks is performed.

On the other hand, the Globus Toolkit is an open source tool that allows the generation of inter-organisational Grids within a secure and transparent environment. It offers basic building tools for data transfer, parallel execution and integration with remote execution policies, among other features.

5.1 Enabling Portability

A distributed Grid infrastructure is, at first glance, an unknown pool of computational resources. It can not be assumed that the execution hosts will have

available the required dynamic libraries that the simulation system depends on, neither the computational nor the system libraries dependences. Therefore, the simulation system should have no requirements of any external library. We have approached this problem by static linking the application, that is, introducing the code from all the dynamic libraries into a single executable with no external library dependences. The MPI message passing layer is also introduced into the executable by static linking with a standard MPICH [9][10] implementation, specially configured to disable shared memory communication between processes in the same node of a cluster, which is known to introduce memory leak problems because of relying on the System V IPC facilities [9].

In addition, all sort of platform-dependent optimised software should not be employed, such as the BLAS or LAPACK libraries implementations for a concrete architecture, as well as platform-dependent compiler optimization flags, such as *-march* or *-mcpu* which allow the compiler to emit specific code for a specific platform. This way, the simulation system will not execute any illegal instruction on the remote machine. Fortunately, traditional compiler optimization flags, i.e. *-O3*, can be used with no risk.

Therefore, it is possible to achieve a self-contained parallel simulation system that can be executed on different Linux versions. Besides, having integrated the MPI communication software, it can be executed in parallel on a remote cluster without depending on the MPI implementation of the execution host. This has been ensured by parallel simulations in a variety of machines of different architectures such as Pentium III, Pentium Xeon and even Intel Itanium 2 with different Linux flavours such as Red Hat Linux Advanced Server, Red Hat 8.0 and Debian GNU/Linux.

This process of adapting the simulation system to the Grid infrastructure results in a weighty executable file that can be lightened by discarding the symbols from its object files. Through compression, with a Lempel-Ziv coding scheme, the executable archive has been reduced, in our case, to a self-contained simulation system of less than 2 MBytes. This self-contained simulator performs on average 2% slower than the optimised counterpart.

It should be pointed out that such a simulator runs on compatibility mode on an Intel Itanium 2 (64 bit) platform and thus, it is up to 8 times slower than on an Intel Pentium Xeon (32 bit). Therefore, we have natively compiled on the Intel Itanium 2 platform in order to achieve comparable execution times on both architectures, and to be able to exploit Itanium Grid execution nodes. This results on two self-contained simulation systems, one for IA-32 and other for IA-64 platforms, an strategy that could be refined to target more architectures.

5.2 Grid Infrastructure

The available Grid infrastructure, shown in Fig. 3, is composed of local resources, belonging to our research group, the High Performance Networking and Computing Group (GRyCAP-UPV), and remote resources from the Distributed Systems Architecture & Security group (ASDS-UCM), at Universidad Complutense de Madrid. Table 2 summarises the main characteristics of the machines.

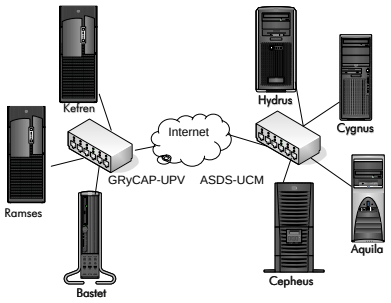


Fig. 3. Computational resources within the Grid infrastructure

The Globus Toolkit version 2.4 [2] has been installed on all the machines of this testbed. Besides, provided that there is no Itanium version of InnerGrid yet, and this software is focused for single-organisational resources, this middleware has only been installed on Ramses and Kefren clusters.

Table 2. Detailed machine characteristics

Machine	Processors	Memory	Job Manager
kefren	10 (2 x Intel Xeon 2.0 Ghz)	1 GByte	pbs, fork
ramses	12 (2 x Intel Pentium III 866 Mhz)	512 MBytes	pbs, fork
bastet	2 x Itanium 2 (900 Mhz)	4 GBytes	fork
hydrus,cygnus	1 x Pentium 4 (2.53 Ghz)	512 MBytes	fork
aquila	1 x Pentium III (666 Mhz)	128 MBytes	fork
cepheus	1 x Pentium III (666 Mhz)	256 MBytes	fork

5.3 Globus Developments

Figure 4 shows a conceptual view of the Grid Computing system developed. The *JobScheduler* is the module responsible for the allocation of simulations to the computational resources. It delegates into a *JobSubmitter* instance, for each simulation, which will be in charge of the proper execution of the task in the resource.

The JobScheduler Module. This module reads an input file with a parametric description of the multiple simulations that form a whole case study. For each simulation, it finds out the best available resource, from a predefined list of machines, by consulting their number of available nodes on the machine, via the Monitoring and Discovery Service (MDS) provided by Globus. Clusters

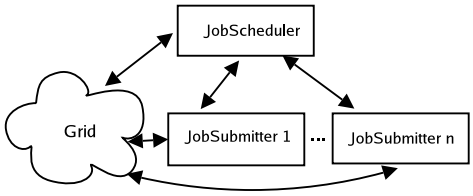


Fig. 4. Scheme of the Grid Computing system developed

with the Globus Resource Allocation Manager (GRAM) Reporter installed report the number of free computing nodes, delegating in the local queue manager (LoadLeveler, PBS, etc). For workstations or sequential PCs, an estimate of the CPU usage during the last minute serves as an indicator of the availability of the resource. This strategy allows to customise a parallel execution to the number of available nodes in the host.

We have included basic quality of service capabilities by specifying the minimum and maximum number of processors on which a parallel simulation can be run. These numbers are thought to increase productivity and to ensure executions with some minimum requirements. Besides, a problem-dependent memory estimator of each simulation prevents the execution of tasks on machines that will otherwise become memory exhausted.

Then, this module selects an appropriate executable based on the architecture of the remote machine. The JobScheduler is also responsible for submitting the unassigned simulations and restarting the failed executions, delegating on an instance of the JobSubmitter module. If no available resources exist, it periodically checks their availability to continue submitting pending tasks.

Given that the amount of data generated by the simulator is quite large (hundreds of MBytes), the GridFTP protocol is employed for all the data transfers as opposed to the GASS (Global Access to Secondary Storage) service. This ensures high performance reliable transfers for high-bandwidth networks.

The JobSubmitter Module. Each instance of this module is in charge of the proper execution of a single simulation. First of all, the input files that the simulation system needs are staged in, via the GridFTP service, to the execution host. Through the Globus native interface, the remote machine is queried about its availability to run MPI jobs, so a serial or parallel execution can be selected. The execution of the simulation is integrated, if configured, with the queue manager of the remote node (PBS, LoadLeveler, etc), thus respecting the execution policies of the organisation.

While the simulation is running on the remote resource, a checkpoint job is periodically submitted by this module, which transfers, if not already done, a compressed image of the checkpoint data generated by the application to the local machine. Thus, the latest checkpoint files always will reside in the submission machine and a failed simulation can be automatically restarted on a new

computational resource. A message digest mechanism ensures that the latest checkpoint data is only transferred once, thus saving bandwidth.

Once the execution has finished, all the result files are compressed, transferred back to the submission node and saved in the appropriate local folder created for this simulation. All the output files in the execution node are deleted, and finally, the JobSubmitter module annotates whether the simulation has finished correctly or not. This information will be used by the JobScheduler module to be able to restart or resume the failed simulations.

5.4 InnerGrid Developments

InnerGrid software has been tested as an alternative, easy-to-use middleware for Grid execution. InnerGrid automatic file staging capabilities, combined with its built-in task scheduler, dramatically simplifies the extra development in order to execute cardiac case studies. We have developed a new module that allows to specify the memory and disk requirements of each simulation. Besides, this module specifies the varying parameters of the study.

Then, a task can be seen as the instantiation of a module, and thus, new tasks, that define the range of variation of the parameters, can be created.

6 Case Study

In order to test the capabilities of the Grid Computing system developed, a real case study has been executed on the available Grid infrastructure.

6.1 Description

Myocardial ischemia is a condition in which oxygen deprivation to the heart muscle is accompanied by inadequate removal of metabolites because of reduced flood. The mechanisms of generation of ventricular tachycardia and ventricular fibrillation (the most mortal of arrhythmias) can be studied using a model of a regionally ischemic cardiac tissue (which would result from the occlusion of a coronary artery) in which certain part of the substrate is ischemic while the surrounding tissue remains in normal conditions [11].

In the ischemic zone, the values of several electrophysiological parameters suffer variations through time as ischemia progresses. Extracellular potassium concentration ($[K^+]_o$), in first place, changes from its normal value (5.4 mmol/L) to a value of 12.5 mmol/L in the first 5 minutes, reaching a plateau for the next 5 minutes of ischemia [12]. In second place, intracellular concentration of ATP ($[ATP]_i$) decreases almost linearly with time from a normal value of around 6.8 mmol/L to 4.6 mmol/L in the first 10 minutes of ischemia, while the intracellular concentration of ADP ($[ADP]_i$) increases from 15 μmol/L to 100 μmol/L in the same time interval [12, 13]. Finally, acidosis reduces the maximum conductance of sodium (I_{Na}) and calcium ($I_{Ca(L)}$) currents to around 75% of its normal values between the fifth and the tenth minute of the ischemic episode [14].

Thus, the four parameters mentioned (which are present in the model of the cardiac action potential) change differently with time during the first 10 minutes of myocardial ischemia. In the simulations presented here, the short-term electrical behaviour of the tissue was simulated in different instants of time after the onset of ischemia. Time was, therefore, the changing parameter of the set of simulations.

The simulated virtual 3D tissue comprised a central ischemic zone, consisting on a 20x20x20-element cube (which represents 2x2x2 mm) in which $[K+]_o$, $[ATP]_i$, $[ADP]_i$, I_{Na} and $I_{Ca(L)}$ changed with time, embedded in a 60x60x60-element cube. The electrophysiological parameters of the tissue that surrounds the central ischemic zone is maintained in their normal values.

This case study will analyse the influence in action potential propagation of different degrees of ischemic conditions that take place from 0 to 10 minutes from the onset of ischemia. Using a time increment of 0.5 minutes, this results in 21 independent parametric simulations that can be executed in a Grid infrastructure. Each execution will perform a simulation during 80 ms with a time step of 10 μs. A supra-threshold stimulus will be applied to all the cells at the bottom plane during the simulation time interval [50, 52] ms. A snapshot of the membrane potential of the tissue cells will be stored every 1 ms during the interval [40, 80] ms, resulting in a total 68 MBytes of raw data. Besides, a total 180 MBytes of RAM is required for the execution of each simulation on a sequential platform.

6.2 Execution Results

For the Globus-based Grid Computing system designed, Table 3 summarises the task distribution in the Grid. As maximum, parallel executions have been limited to a quarter the total available processors of the remote resource, implementing a polite policy that allows multiple concurrent simulations. The minimum number of processors where set to one, thus allowing serial executions. In the table, an entry like 3 (5 p.) indicates that three simulations were performed with five processors each one.

The machines Bastet and Cepheus do not appear in the table because they were heavily overloaded during the test and thus, the scheduler never chose them. Besides, the machine Aquila did not have enough RAM to host a simulation and thus, the scheduler did not considered it for execution.

Table 3. Distribution of the simulations in the testbed, for each machine. The number in parentheses indicates the number of processors involved in the execution

Machine	Simulations	Machine	Simulations
Kefren	4 (4 p.), 3 (3 p.), 2 (2 p.) 2 (1 p.)	Hydrus	2 (1 p.)
Ramses	1 (6 p.), 3 (5 p.), 1 (4 p.), 1 (1 p.)	Cygnus	2 (1 p.)

The execution of the whole case study lasted for 10.27 hours in the Grid deployment. It can be seen that the scheduler dynamically assigned to each machine a number of simulations proportional to its computational power, thus resulting in a scheduling policy that takes into account Grid computers with little computational resources.

The execution of the case study via the InnerGrid middleware lasted for 22.57 hours, distributing the tasks among the nodes of the two clusters. Executing the same case study in a traditional sequential manner in one node of the cluster Kefren required 81.1 hours. Finally, performing a High Performance Computing approach, executing each simulation with 4 processors in the same cluster (thus allowing two concurrent executions), required a total 11.9 hours.

While InnerGrid seems more appropriate to take advantage of idle computers in single-organisational Grids, the Globus Toolkit is more focused on running on dedicated resources from different organisations. Therefore, a Globus-based solution is much more appropriate for the cardiac electrical activity simulation problem, as it offers the possibility to access distant computational resources in a transparent manner.

6.3 Case Study Results

Figure 5 summarises the results obtained in the case study. It represents the membrane potential of the cells of a vertical tissue slab at the center of the cube, for three different ischemic conditions and at four simulation time instants. Ischemic conditions at 0 minutes introduce no perturbation in action potential propagation, while more severe degrees of ischemia provoke a slowdown in propagation within the tissue area affected.

Figure 6 offers a three-dimensional representation of the membrane potential of all the cells of the tissue at simulation time 59 ms, for three different degrees of ischemia. Again, it can be seen that the ischemic zone reduces the velocity of action potential propagation, which is more severe as the ischemia progresses in time.

7 Conclusions

This paper presents an efficient parallel implementation for the simulation of the action potential propagation on a three-dimensional monodomain ventricular cardiac tissue. The use of a High Performance Computing-based model has reduced the simulation times from days to few hours. Besides, larger 3D tissues can be simulated during longer time by only employing more processors, thus enlarging the global available memory.

In addition, to harness resource-starved cardiac case studies, a Globus-based Grid Computing system has been developed, what allows the integration of MPI parallel executions on multiprocessor machines within a Grid infrastructure. The system features important capabilities such as self-contained executable, dependencies migration, data compression, cross-linux portability, as well as the integration of parallel executions in the Grid.

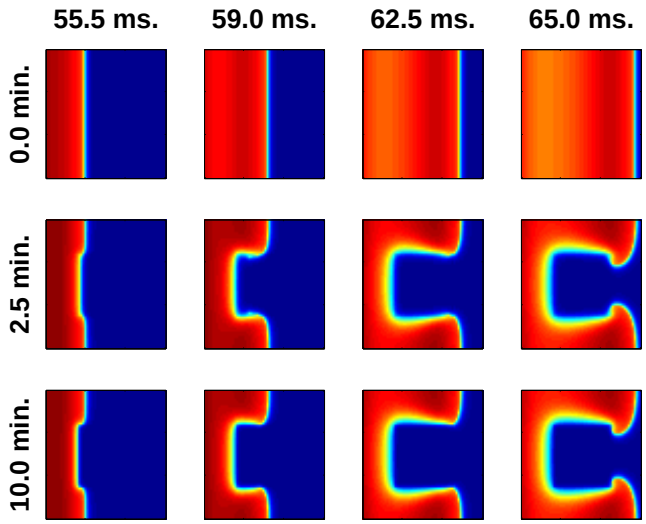
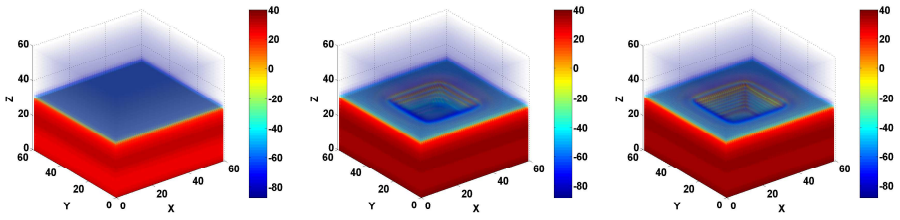


Fig. 5. Membrane potential at four simulation time instants under three degrees of ischemia. Propagation takes place from left to right. Colourbar is the same as shown in Fig. 6



(a) Ischemia after 0 min.

(b) Ischemia after 2.5 min.

(c) Ischemia after 10 min.

Fig. 6. Three-dimensional representation of the membrane potential of the tissue at simulation time instant 59 (ms.)

Besides, InnerGrid commercial product has been tested as an alternative middleware for creating single-organisational Grids. As part of our work, a new InnerGrid module has been developed which allows varying several parameters and managing the execution of the parametric tasks from a web environment.

As High Performance Computing techniques are responsible for *speedup*, Grid Computing is responsible for *productivity*. The integration of both computational strategies has represented to be a key combination in order to enhance productivity when executing cardiac case studies.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications **15** (2001) 200–222
2. Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. Intl. J. Supercomputer Applications **11** (1997) 115–128
3. GridSystems S.A.: InnerGrid Nitya Technical Specifications. (2003)
4. Luo, C.H., Rudy, Y.: A Dynamic Model of the Cardiac Ventricular Action Potential. I Simulations of Ionic Currents and Concentration Changes. Circulation Research **74** (1994) 1071–1096
5. Vigmond, E.J., Aguel, F., Trayanova, N.A.: Computational Techniques for Solving the Bidomain Equations in Three Dimensions. IEEE Transactions on Biomedical Engineering **49** (2002) 1260–1269
6. Balay, S., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, M., McInnes, L.C., Smith, B.F., Zhang, H.: PETSc User Manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory (2002)
7. Amestoy, P.R., Duff, I.S., L'Excellent, J.Y., Koster, J.: MUltifrontal Massively Parallel Solver (MUMPS Version 4.3) User's Guide. (2003)
8. Karypis, G., Kumar, V.: METIS. A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes and Computing Fill-reducing Orderings of Sparse Matrices. University of Minnesota. Version 4.0. (1998)
9. Gropp, W.D., Lusk, E.: User's Guide for MPICH, a Portable Implementation of MPI. Mathematics and Computer Science Division, Argonne National Laboratory. (1996)
10. Gropp, W., Lusk, E., Doss, N., Skjellum, A.: A High-Performance, Portable, Implementation of the MPI Message Passing Interface Standard. Parallel Computing **22** (1996) 789–828
11. Coronel, R.: Heterogeneity in Extracellular Potassium Concentration During Early Myocardial Ischaemia and Reperfusion: Implications for Arrhythmogenesis. Cardiovasc. Res. **28** (1994) 770–777
12. Weiss, J.N., Venkatesh, N., Lamp, S.T.: ATP-sensitive k^+ Channels and Cellular k^+ Loss in Hypoxic and Ischaemic Mammalian Ventricle. J. Physiol. **447** (1994) 649–673
13. Ferrero (Jr.), J.M., Saiz, J., Ferrero, J.M., Thakor, N.: Simulation of Action Potentials from Metabolically Impaired Cardiac Myocytes. Role of ATP-sensitive k^+ Current. Circ. Res. **79** (1996) 208–221
14. Yatani, A., Brown, A.M., Akaike, N.: Effects of Extracellular pH on Sodium Current in Isolated, Single Rat Ventricular Cells. J. Membr. Biol. **78** (1984) 163–168