A COMPUTATIONAL FRAMEWORK FOR CARDIAC MODELING BASED ON DISTRIBUTED COMPUTING AND WEB APPLICATIONS

D. M. S. Martins[1,2], F. O. Campos[1], L. N. Ciuffo[1], R. S. Oliveira[1], R. M. Amorim[1], V. F. Vieira[1], N. F. F. Ebecken[2], C. B. Barbosa[1], R. Weber dos Santos[1]

[1] Department of Computer Science , Universidade Federal de Juiz de Fora,
Juiz de Fora, Minas Gerais, Brazil
rodrigo@dcc.ufjf.br
[2] Department of Computer Science , Universidade Federal do Rio de Janeiro,
Rio de Janeiro, Rio de Janeiro, Brazil
nelson@ntt.ufrj.br

Abstract. Cardiac modeling is here to stay. Computer models are being used in a variety of ways and support the tests of drugs, the development of new medical devices and non-invasive diagnostic techniques. Computer models have become valuable tools for the study and comprehension of the complex phenomena of cardiac electrophysiology. However, the complexity and the multidisciplinary nature of cardiac models still restrict its use to a few specialized research centers in the world. We propose a computational framework that provides support for cardiac electrophysiology modeling. This framework integrates different computer tools and allows one to bypass many complex steps during the development and use of cardiac models. The implementation of cardiac cell models is automatically provided by a tool that translates models described in CellML language to executable code that allows one to manipulate and solve the models numerically. The automatically generated cell models are integrated in an efficient 2-dimensional parallel cardiac simulator. The set up and use of the simulator is supported by a user-friendly graphical interface that offers the tasks of simulation configuration, parallel execution in a pool of connected computer clusters, storage of results and basic visualization. All these tools are being integrated in a Web portal that is connected to a pool of clusters. The Web portal allows one to develop and simulate cardiac models efficiently via this user-friendly integrated environment. As a result, the complex techniques and the know-how behind cardiac modeling are all taken care of by the web distributed applications.

1 Introduction

The phenomenon of electric propagation in the heart comprises a set of complex non-linear biophysical processes. Its multi-scale nature spans from nanometre processes such as ionic movements and protein dynamic conformation, to centimetre phenomena such as whole heart structure and contraction. Computer models [1,2] have become valuable tools for the study and comprehension of such complex phenomena, as they allow different information acquired from different physical scales and experiments to be combined in order to generate a better picture of the whole system functionality.

Not surprisingly, the high complexity of the biophysical processes translates into complex mathematical models. The modern cardiac electrophysiology models are described by non-linear systems of partial differential equations with millions of variables and hundreds of parameters. Whereas the setup process of the simulations is time consuming and error prone, the numerical resolution demands high performance computing environments. In spite of the difficulties, the benefits and applications of these complex models justify their use. Computer models have been used during the tests of drugs [3], development of new medical devices [4], and of new techniques of non-invasive diagnosis [5] for several heart diseases.

We propose a computational framework that provides support for cardiac electrophysiology modelling. A web portal architecture which combines server and applications is presented in Figure 1.

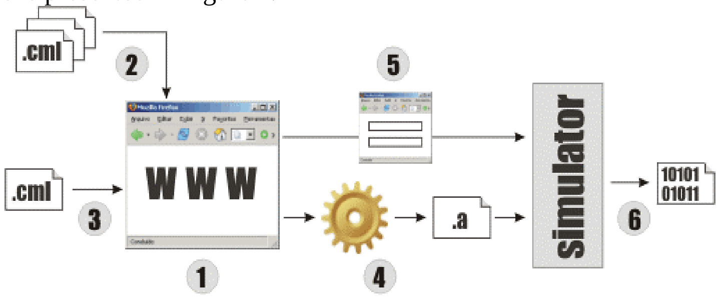


Fig. 1. The high level architecture

Through a public website (1), a user can select a biological model previously stored in the system (2), or submit its own model described in CellML meta-language (3), which has recently emerged as an international standard for the description of cell models [6]. Once established the biological model to be used, a compiler for CellML [6] will generate parallel C code based on the Message Passing Interface library (MPI [7]) (4). The result of the compilation is used as input by the simulator software which we have previously developed [8]. At the same time, the user can type the parameters in an electronic form to configure the initial states and conditions for the simulation (5). These parameters are used as simulator's data input. The simulations run on a pool of clusters and generate binary files in the end of the process (6) which can be downloaded and visualized (Figure 2).

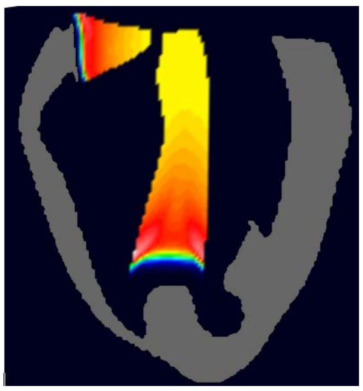


Fig. 2. Visualization of simulated results showing an electrical wave that propagates through the ventricles.

The tools described above (XML based code generator, parallel cardiac simulators, graphical user interface environments and the web portal) provide a user-friendly environment for cardiac simulation. The complex techniques and know-how that are behind cardiac simulations, such as parallel computing, advanced numerical methods, visualization techniques and even computer code programming, are all hidden behind the integrated and easy-to-use web based framework.

The next sections describe the details of each of these components.

2 AGOS Tool

There are two basic components in mathematical models of cardiac electric propagation: the cell model and the tissue model. The first component models the flow of ions across the cell membrane as first proposed by Hodgkin and Huxley [2] in their work on nerve cells. This component typically comprises of a system of Ordinary Differential Equations (ODEs). The Second component is an electrical model for the tissue that describes how currents from one region of a cell membrane interact with the neighborhood. This component is represented by a Partial Differential Equation (PDE). In this section, we present an on-line tool, the AGOS tool, aimed to help researchers in the development and solution of cell models or any other scientific model based on systems of ODEs. Special computational tools for handling the second component (the PDEs) are covered in the next sections.

AGOS stands for API (Application Program Interface) Generator for ODE Solution. Through its use one can submit a meta-model file to automatically generate a C++ API for solving first-order initial-value ODE systems.

The input data is a CellML [6] or a Content MathML [9] file, i.e., XML-based languages. CellML is an open-source mark-up language used for defining mathematical and electrophysiological models of cellular function. MathML is a W3C standard for describing mathematical notation. A CellML file includes Content MathML to provide both a human- and computer-readable representation of mathematical relationships of biological components [10]. Therefore, the AGOS tool allows the submission of a complete CellML file or just its MathML subset.

Once submitted, the XML file is translated to an API. The generated API is an object oriented C++ code. Functions are created for system initialization (initialization of parameters like the number of iterations, discretization interval and initial values of variables), numerical solution (via Explicit Euler scheme) and results storage. In addition, the API offers public reflexive functions used, for example, to restore the number of variables and their names. These reflexive functions allow the automatic creation of model-specific interfaces. This automatic generated interface enables one to set any model initial condition or parameter, displaying their actual names, as documented in the CellML or MathML input file. The AGOS tool is available at (www.fisiocomp.ufjf.br), from where it is possible to download the API source-code. AGOS can also be used online via a web application (see section 4), which uses the generated API to solve ODE systems and visualize their results.

In the next section, we present how the XML code is translated to C++ code.

2.1 The Translator

The AGOS application was implemented in C++ and makes use of basic computer structure and algorithms in order to capture the variables, parameters and equations, i.e. the ODE conceptual elements, that are embedded in a MathML file and translate these to executable C++ code, i.e. the AGOS API. The translator tool comprises of three basic components: a Preprocessor for XML format, an Extractor of ODE conceptual elements, and a Code Generator. The components are organized as a pipeline. The Preprocessor reads an XML-based file (MathML or CellML) and extracts the content into an array of tree data structures. Every tree of this array is processed by the ODE extractor that identifies the ODE elements and stores them in appropriate data formats. At the end of the pipeline, the Code Generator combines the extracted information to a code template and generates the AGOS API. The adopted strategy for code generation is largely based on code templates. The syntactical structure of code templates is described using formal grammar notation. Details related to the AGOS API and to the translator are documented in the AGOS manuals that can be found at [11].
The MathML description language uses a prefix format, i.e., the operators precedes the operands. The translator goal is achieved via the creation of a structure that supports easy identification of the operands and operators. AGOS converts the XML embedded equations in a tree-like structure. We briefly illustrate the translator tasks via a simple example. Consider the following equation:

$$t = 6.0 + \sqrt[n]{a} \qquad (1)$$

The corresponding Content MathML code and the generated tree are presented in Figure 3.

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply><eq/>
    <ci> t </ci>
    <apply><plus />
      <cn> 6.0 </cn>
      <apply><root/>
        <degree><ci> n </ci></degree>
                      <ci> a </ci>
      </apply>
    </apply>
  </apply>
</math>
```

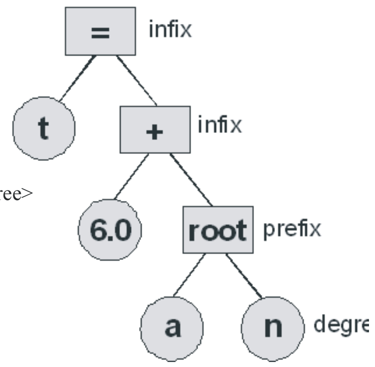


Fig. 3. Content MathML code and the extracted tree structure.

The tree nodes contain information about each operand and operator, besides the equation type (if it is a differential equation or an algebraic one). The translator uses this information to include the mathematical code in the right place in the API. Using a search in depth, the following code is generated: “t = (6.0 + pow(a, 1.0/n));”.

3 The Parallel Cardiac Simulator

The set of Bidomain equations [12] is currently one of the most complete mathematical models to simulate the electrical activity in cardiac tissue:

$$\nabla \cdot (\sigma_i \nabla \phi_i) = \chi \left(C_m \frac{\partial \phi}{\partial t} + f(\phi, \vec{n}) \right) \qquad \textbf{(2)}$$

$$\nabla \cdot (\sigma_e \nabla \phi_e) = -\chi \left(C_m \frac{\partial \phi}{\partial t} + f(\phi, \vec{n}) \right) \qquad \textbf{(3)}$$

$$\frac{\partial \vec{n}}{\partial t} = g(\phi, \vec{n}) \qquad \textbf{(4)}$$

, where ϕ_e is the extracellular potential, ϕ_i the intracellular potential and ϕ is the transmembrane potential. Eq. 4 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 Eq. 4 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. 3x3 symmetric matrices that vary in space and describe the anisotropy of the cardiac tissue. Cm 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 (2)–(4) at every time step is via the operator splitting technique [13]-[15]. The numerical solution reduces to a modular 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 (2)–(4) using the operator splitting technique (see [16] for more details) we get the following numerical scheme:

$$\varphi^{k+1/2} = (1+\Delta t A_i)\varphi^k + \Delta t A_i(\varphi_e)^k; \tag{5}$$

$$\begin{aligned} \varphi^{k+1} &= \varphi^{k+1/2} - \Delta t f\left(\varphi^{k+1/2}, \vec{\xi}^{\,k}\right)/Cm \\ \vec{\xi}^{\,k+1} &= \vec{\xi}^{\,k} + \Delta t g\left(\varphi^{k+1/2}, \vec{\xi}^{\,k}\right); \end{aligned} \tag{6}$$

$$(A_i + A_e)(\varphi_e)^{k+1} = -A_i \varphi^{k+1}; \tag{7}$$

where φ^k, φ_e^k and ξ^k discretizes ϕ, ϕ_e and η at time k Δt; A_i and A_e are the discretizations for $\nabla.\sigma_i\nabla$ and $\nabla.\sigma_e\nabla$, respectively. Spatial discretization was done via the Finite Element Method using a uniform mesh of squares and bilinear polynomials as previously described in [16].

Steps (5), (6) and (7) are solved as independent systems. Nevertheless, (5), (6) and (7) are still computationally expensive. One way of reducing the time spent on solving these equations is via parallel computing.

3. 1 The 2-dimensional Parallel Cardiac Simulator

A solution for the Bidomain model was implemented in parallel using the MPI and PETSc [17] libraries. PETSc provides a suite of data structures and routines for the scalable solution of applications modeled by PDEs. The nonlinear ODE system was solved via the explicit forward-Euler scheme (see section 3.2 for more details). The PDEs are the most computationally expensive portion of the model and thus need a more robust algorithm. The Conjugate Gradient (CG) method combined with an appropriate preconditioner has become a standard choice for an iterative solver and was applied for the solution of the elliptic and parabolic PDEs.

The CG was parallelized via linear domain decomposition. The spatial rectangular domain was decomposed into *nproc* nonoverlapping domains of equal size, where *nproc* was the number of processors involved in the simulation. For the 2-D problem, the slice was made in the *y* direction.

3. 2 Integration with AGOS API

As described in Section 2, one can select or submit a biological model through a public web site. This model is compiled into a scalable (parallel) shared library, which can be used by the simulator.

The shared library is dynamic, thus the Simulator does not need to be compiled for every new cell model. Therefore, when it is necessary to solve the nonlinear system of ODEs, i.e. during the step described by equation (6), the Simulator makes a call to the functions of the automatic generated API library on run time, the functions are loaded to memory and executed.

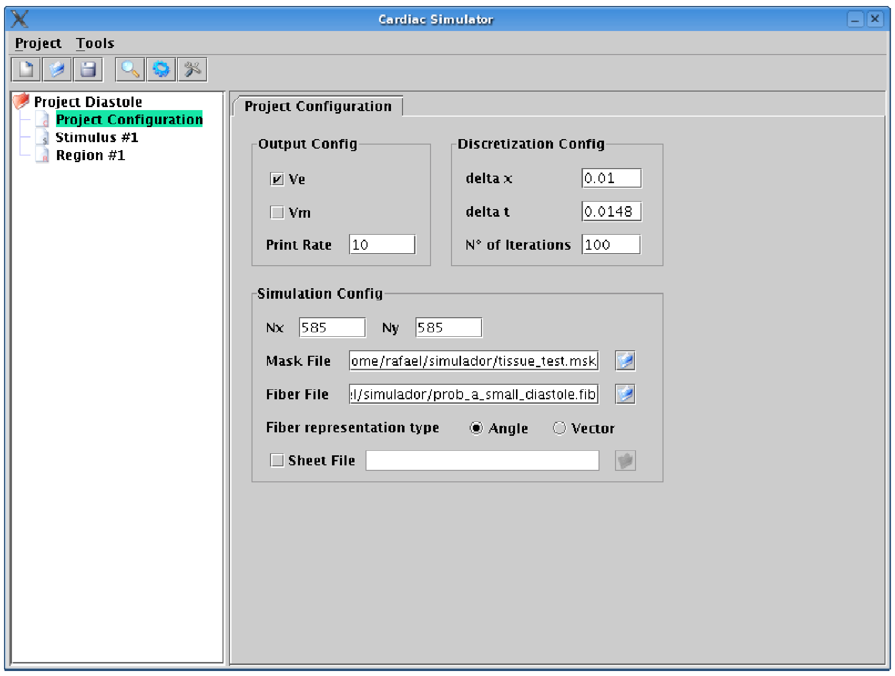
The parallelization is easily obtained: each processor is responsible for a rectangular domain (nx X ny/nproc) that has nx ny /nproc ODE systems associated with. These are independent systems. The solution of these systems does not involve communication.

3. 3 Simulator Graphic User Interface (GUI)

In order to provide an easy way for setting up simulations, a Java GUI was developed and it is here briefly presented.

The GUI allows the creation, set up, execution and visualization of simulations. First, one must create a project and set several parameters for simulating the spread of cardiac electric activity. Then, the parallel simulation can be started and its results will be stored in a folder that has the same name of the project. Visualization of the results can also be initiated via the GUI. Currently the visualization is done via an interface to FLOUNDER, a free software developed at the Calgary University [18].

Once the project is created, it is possible to select which variables should be saved for later analyzes (intra-, transmembrane, extracellular potentials, etc.). Parameters as time and space discretization, size of the bidimensional portion of tissue and number of iterations can be set. Via the selection of input files one defines the geometry of the model, cardiac fiber and sheet orientations. Stimuli type, location and intensity can also be configured. Figure 4 shows an example of a simulation set up using the simulator GUI.



Output Config
- [✓] Ve
- [ ] Vm
- Print Rate: 10

Discretization Config
- delta x: 0.01
- delta t: 0.0148
- N° of Iterations: 100

Simulation Config
- Nx: 585
- Ny: 585
- Mask File: ome/rafael/simulador/tissue_test.msk
- Fiber File: :l/simulador/prob_a_small_diastole.fib
- Fiber representation type: [✓ Angle] Vector
- [ ] Sheet File

Fig. 4. Simulation set up using the GUI

3. 4 The Cluster

The set up of a parallel environment based on commodity network and desktop computers can be a complex task. In order to set up a Linux cluster for scientific computing, several software packages are necessary for tasks such as installation, configuration and management. Fortunately, there are some popular kits that support these tasks.

The cluster used in this work [11], is based on NAPCI Rocks architecture and it is made up of eight nodes, Athlon 64 3000+ with 2 GB of RAM, connected by a fast Gigabit Ethernet switching device. Rocks [19] is a collection of open source software integrated to the Red Hat Linux which aims at building high performance clusters.

For monitoring the cluster status, Rocks provides a tool named Ganglia, which is a scalable distributed monitoring system for high-performance computing systems. Ganglia allows the cluster administrator to visualize historical monitoring information for cluster, host, and metric trends over different time granularities ranging from minutes to years. It generates graphics that present the historical trends of metrics versus time. Typical and useful graphics include information on network bandwidth utilization and CPU load for the whole cluster as well as for each individual node. All the monitoring in Ganglia can be done via a web browser.

In addition, Rocks is integrated to Sun Grid Engine (SGE). SGE schedules the jobs submitted to the cluster to the most appropriate nodes, based on management policies previously defined by the cluster administrator. SGE is integrated with Message Passing Interface and Parallel Virtual Machine and allows users to run parallel jobs based on these libraries. Any number of different parallel environment interfaces can be configured concurrently. Sun Grid Engine also provides dynamic scheduling and job migration via checkpoints, i.e., the procedure of storing the state of an active process. A graphical interface called QMON provides for easy control and configuration of all SGE capabilities.

3. 5 Simulation Example

In this section we present an example of the use of the environment discussed in the preceding sections. We simulate the cardiac electric propagation on a 2-dimensional cut of the left ventricle obtained during the cardiac systole phase by the resonance magnetic technique of a healthful person. After segmenting the resonance image, a two-dimensional mesh of 769 X 769 points is generated, that models the cardiac tissue, blood and torso. All bidomain parameters were taken from [20]. The cardiac tissue conductivity values have been set to: σ_{il} = 3 mS/cm, σ_{it} = 0.31 mS/cm, σ_{el} = 2 mS/cm and σ_{et} = 1.35 mS/cm, where i(e) denotes intracellular (extracellular), l(t) and stands for longitudinal (transversal) to the fiber orientation. The capacitance per unit area and the surface area-to-volume ratio are set to 1 mF/cm2 and 2000 /cm, respectively. The interface between cardiac tissue and bath is modeled as described in [21]. All the other boundaries are assumed to be electrically isolated. The spatial and temporal discretization steps of the numerical model are set to 150 µm and 10 µs, respectively. The simulation was carried out for 20 ms 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 [22]. The explicit Euler implementation described by equation (6) was generated automatically by AGOS, based on a CellMl model description downloaded from the CellML repository [6]. The linear system associated to the parabolic part of the bidomain formulation, see equation (5), is solved with the Conugate Gradient (CG) method and ILU (Incomplete LU factorization with zero fill-in) preconditioner. The linear system associated to the elliptic part, equation (7), dominates computation and is solved with CG and a parallel Algebraic Multigrid preconditioner. In this work we adopted the parallel AMG code BoomerAMG [23] with its Falgout-coarsening strategy.

Figure 5 shows the simulation result overlapped to the resonance image. The color-coded image represents the transmembrane potential distribution for a certain time instant.

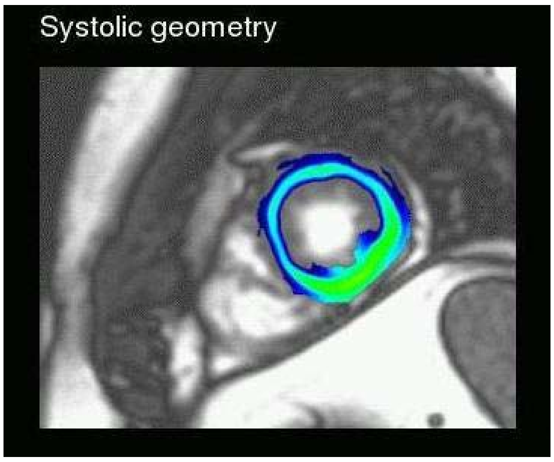


Fig. 5. Simulated electrical wave propagation overlapped to the Resonance Image.

The simulation was run using one, two, four and eight processors. As presented in Figure 6, when running the simulation on eight processors the relative speedup (execution time using 1 processor / execution time with n processors) is near 5. The execution time drops from near 5 hours when running with 1 processor to less than 1 hour running with 8 processors. Linear speedups were not achieved. This is mainly due to communication overhead and to the Multigrid Preconditioner adopted. The direct method used in the coarsest grid of the preconditioner is not parallelized and thus limits scalability. Nevertheless, the results indicate the importance and benefits of cluster computing for cardiac electrophysiology modeling.

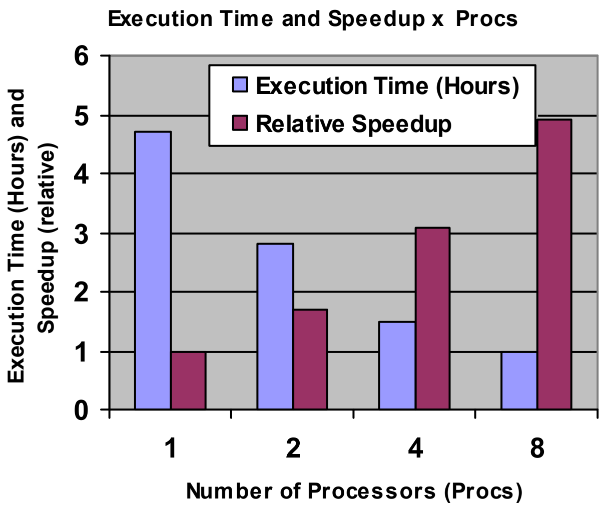


Fig. 6. Parallel speedup and execution time in hours.

4. The Web Portal: goals, current status and future work

The Web portal is under development and has three main goals:

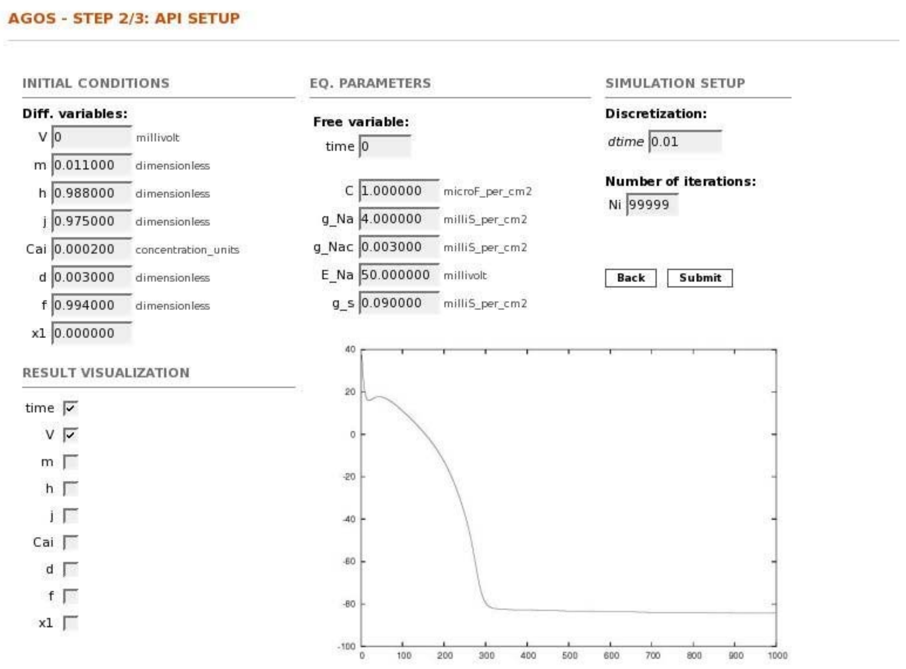
1- To popularize the technology of cardiac simulation. The tools described above (XML based code generator, parallel cardiac simulators and user interface environments) are being combined in a single Web application which has access to a pool of clusters. The web portal supports the development and simulation of cardiac models efficiently via this user-friendly integrated environment. The complex techniques and know-how that are behind cardiac simulations, such as parallel computing, advanced numerical methods, visualization techniques and even computer code programming, are all taken care by the integrated and easy-to-use web application. Online tutorials [11] instruct the users on how to make efficient use of the integrated environment.

2- To promote the share of the computational resources among different research centers. The web application under development will allow users to execute their simulations on a pool of clusters made of clusters residing on different research centers. This will bring a parallel environment to those that do not have access to it as well as increase the computational power of the participating centers. Currently, there are three small clusters being integrated to the Web Portal (4-node cluster from Lanec-UFSJ (Neuroscience Laboratory), 5-node cluster from Labma-UFRJ (Applied Mathematics Laboratory), and the 8-node cluster from FISIOCOMP-UFJF (Computational Physiology Laboratory). This integration will be done with the Sun Grid Engine, which provides the mechanisms required for grid computing.

3- To promote the development of cardiac modeling. The integration of the above mentioned tools in a single web portal will speed up the development of new and more realistic electro-physiological models of the heart and further integrate different research centers, promoting international collaborations.

Currently the AGOS tool is fully operational and integrated to the Web Portal. After registration, the user is granted a new account and is able to create, manage, execute and store the results of simulation projects. Associated to each registered researcher there is a folder in the server's hard disk. This folder contains one subfolder for each created project, with all the input and output files generated by the AGOS application. Those files can be downloaded and visualized at anytime. In addition, the researcher may modify the API parameters to generate new PDF and PS graphics. One of the portal screenshots is shown in the Figure 7.

We are using JSP [24] and Struts framework [25] and some web pages also use PHP [26] language. All data is being stored in a MySQL [27] database and the web-site runs in an Apache Tomcat server [28]. The simulator GUI described in section 3.3 is currently being integrated to the web portal. Combined with AGOS, the parallel cardiac simulator and the SGE grid computing tool, this easy-to-use computational framework will efficiently support cardiac modeling in distributed environments.



AGOS - STEP 2/3: API SETUP

INITIAL CONDITIONS

Diff. variables:

V	0	millivolt
m	0.011000	dimensionless
h	0.988000	dimensionless
j	0.975000	dimensionless
Cai	0.000200	concentration_units
d	0.003000	dimensionless
f	0.994000	dimensionless
x1	0.000000	

EQ. PARAMETERS

Free variable:

time 0

C	1.000000	microF_per_cm2
g_Na	4.000000	milliS_per_cm2
g_Nac	0.003000	milliS_per_cm2
E_Na	50.000000	millivolt
g_s	0.090000	milliS_per_cm2

SIMULATION SETUP

Discretization:

dtime 0.01

Number of iterations:

Ni 99999

RESULT VISUALIZATION

time [✓]
V [✓]
m
h
j
Cai
d
f
x1

Fig. 7. Web portal usage example

5. Conclusion

In this work we presented a computational framework that supports cardiac electrophysiology modeling. The framework is made of different components and technologies and aim on simplifying the development and use of cardiac models. The combination of an XML based automatic code generator, parallel cardiac simulators, graphical user interface environments and a web portal provides an user-friendly environment for cardiac simulation. The complex techniques and know-how that are behind cardiac modeling, such as parallel computing, advanced numerical methods, visualization techniques and even computer code programming are all hidden behind the integrated and easy-to-use web based framework

Acknowledgements

This work was supported by the Brazilian Ministry of Science and Technology, CNPq (process 506795/2004-7).

References

[1] HENRIQUEZ C. S. (1993): ‘Simulating the electrical behavior of cardiac tissue using the bidomain model’, Crit Rev. Biomed. Eng, 21, 1-77

[2] HODGKIN A. L., and HUXLEY A. F. (1952): ‘A quantitative description of membrane current and its application to conduction and excitation in nerve’, J. Physiol., 117, 500-544

[3] GIMA K., and RUDY Y. (2002): ‘Ionic current basis of electrocardiographic waveforms: a model study’, Circ. Res., 90, 889-896

[4] SANTOS R. W. D., STEINHOFF U., HOFER E., SANCHEZ-QUINTANA D., and KOCH H. (2003): ‘Modelling the electrical propagation in cardiac tissue using detailed histological data’, Biomedizinische Technik. Biomedical Engineering, 48, 476-478

[5] SANTOS R. W. D., KOSCH O., STEINHOFF U., BAUER S., TRAHMS L., and KOCH H. (2004): ‘MCG to ECG source differences: measurements and a 2D computer model study’, Journal Of Electrocardiology, 37 Suppl

[6] CellML biology, math, data, knowledge., Internet site address: http://www.cellml.org/

[7] MPI (Message Passing Interface), Internet site address: http://www.mpi-forum.org/

[8] SANTOS R. W. D., PLANK G., BAUER S., and VIGMOND E. J. (2004): ‘Parallel Multigrid Preconditioner for the Cardiac Bidomain Model’, IEEE Trans. Biomed. Eng., 51(11), 1960-1968

[9] Mathematical Markup Language (MathML) Version 2.0 (Second Edition), Internet site address: http://www.w3.org/TR/MathML2/

[10] LLOYD C. M., HALSTEAD M. D. B., and NIELSEN P. F. (2004): ‘cellML: its future, present and past’, in Biophysics & Molecular Biology, 85, 433-450

[11] FISIOCOMP: Laboratory of Computational Physiology, internet site address: http://www.fisiocomp.ufjf.br/

[12] SEPULVEDA N. G., ROTH B. J., and WIKSWO Jr. J. P. (1989): ‘Current injection into a two-dimensional anistropic bidomain’, Biophysical J., 55, 987-999

[13] VIGMOND E., AGUEL F., and TRAYANOVA N. (2002): ‘Computational techniques for solving the bidomain equations in three dimensions’, IEEE Trans. Biomed. Eng., 49, 1260-9

[14] SUNDNES J., LINES G., and TVEITO A. (2001): ‘Efficient solution of ordinary differential equations modeling electrical activity in cardiac cells’, Math. Biosci., 172, no. 2, 55-72

[15] KEENER J., and BOGAR K. (1998): ‘A numerical method for the solution of the bidomain equations in cardiac tissue’, Chaos, 8, no. 1, 234-241

[16] SANTOS R. W. D., PLANK G., BAUER S., and VIGMOND E. Ej (2004): ‘Preconditioning techniques for the bidomain equations’. Lecture Notes In Computational Science And Engineering, 40, 571-580

[17] PETSc: Portable, Extensible Toolkit for Scientific Computation, Internet site address: http://www-unix.mcs.anl.gov/petsc/petsc-as/

[18] Calgary University, Internet site address: http://www.ucalgary.ca/

[19] NAPCI Rocks, Internet site address: http://www.rocksclusters.org/Rocks/

[20] Muzikant,A.L. and C.S.Henriquez. (1998): ‘Validation of three-dimensional conduction models using experimental mapping: are we getting closer?’ Prog. Biophys. Mol. Biol. 69:205-223.

[21] Krassowska,W. and J.C.Neu. (1994): ‘Effective boundary conditions for syncytial tissues.’ IEEE Trans. Biomed. Eng 41:143-150.

[22] ten TUSSCHER K. H. W. J., NOBLE D., NOBLE P. J., and PANFILOV A. V. (2004): ‘A model for human ventricular tissue’, J. Physiol., 286, 1573-1589

[23] Henson V. E., and Yang U. M. (2000): ‘BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner. ’ Technical Report UCRL-JC-139098, Lawrence Livermore National Laboratory

[24] Sun Microsystems, Internet site address: http://java.sun.com/products/jsp/

[25] Apache Software Foundation, internet site address: http://struts.apache.org/

[26] The PHP Group, Internet site address: http://www.php.net/

[27] MySQL AB., Internet site address:http://www.mysql.com/

[28] The Apache Jakarta Project, Internet site address: http://jakarta.apache.org/tomcat/