35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

Code Generator for Distributed Parameter Biological Model Simulation with PDE Numerical Schemes

Florencio Rusty Punzalan, Yoshiharu Yamashita, Masanari Kawabata, Takao Shimayoshi,
Hiroaki Kuwabara, Yoshitoshi Kunieda and Akira Amano

Abstract— **The physiological simulation at the tissue and organ level typically involves the handling of partial differential equations (PDEs). Boundary conditions and in cases like pharmacokinetics, distributed parameters add to the complexity of the PDE solution. These factors make most PDE solutions and their corresponding program codes tailored for specific problems. We propose a general approach for handling PDEs in computational models using a replacement scheme for discretization. This method allows for the handling of the different PDE types. The replacement scheme involves substituting all the partial differential terms with the numerical solution equations. Once the model equations are discretized with the numerical solution scheme, instances of the equations are generated to undergo dependency analysis. The result of the dependency analysis is then used to determine the simulation loop structure and generate the program code.**

I. INTRODUCTION

The lumped parameter system has been widely used for the mathematical modelling of physical and biological functions due to its ease of modelling and high analyticity [1]. However, organ function simulation and analysis increasingly requires a distributed parameter system because it necessitates distribution of physiological structures and spatial localization of intracellular materials.

Biological function model description languages such as SBML [2], CellML [3] and PHML [4] and their wide-ranging sample models are examples of a lumped parameter system. On the other hand, FieldML [5] and FML [6] are description languages capable of describing a distributed parameter system. However, these languages are not versatile for hybrid lumped-distributed parameter physiological systems. Their use and interface are also limited to a single tool or implementation like OpenCMISS and Chaste.

The structure of simulation programs of a lumped parameter system described by ordinary differential equations (ODEs) is relatively homogeneous. However, a distributed parameter system described by partial differential equations (PDEs) can lead to various solutions depending on the problem's initial value, boundary condition, spatial discretization and equation form. The complexity and size of the biological function models in distributed parameter systems make it difficult for life scientists to implement and generate the necessary program code.

F.R. Punzalan, M. Kawabata and A. Amano are with the Department of Bioinformatics, Ritsumeikan Univerisity, Shiga-ken, 525-8577, Japan (phone:+81-77-561-2584; e-mail:floren@fc.ritsumei.ac.jp)

Y. Yamashita, H. Kuwabara and Y. Kunieda are with the Department of Informatics, Ritsumeikan Univerisity, Shiga-ken, 525-8577, Japan)

T. Shimayoshi is with ASTEM RI, Kyoto, 600-8813, Japan

The authors would like to thank the Japan Ministry of Education, Culture, Sports, Science and Technology (MEXT) for the Grant-in-Aid for Scientific Research (Kakenhi) funding

In this study, we propose a code generation system that can automatically generate program codes for distributed parameter system simulations described by PDEs. Through this system, various numerical solution methods can be used to discretize the model equations in an organ-level simulation. It allows users to specify the finite difference scheme that they want to use in their simulation.

II. METHOD

A. Numerical Schemes for Partial Differential Equations

A partial differential equation is defined as the relationship between a function, $\boldsymbol{v}(d_1, d_2, \ldots, d_{nd})$, and the partial derivative of its independent variables $(d_1, d_2, \ldots, d_{n_d})$ as shown by

$$\boldsymbol{\mathcal{F}}\left(\boldsymbol{v}, d_1, d_2, \ldots, d_{n_d}, \frac{\partial \boldsymbol{v}}{\partial d_1}, \frac{\partial^2 \boldsymbol{v}}{\partial d_1^2}, \ldots, \frac{\partial \boldsymbol{v}}{\partial d_2}, \frac{\partial^2 \boldsymbol{v}}{\partial d_2^2}, \ldots, \frac{\partial \boldsymbol{v}}{\partial d_{n_d}}, \frac{\partial^2 \boldsymbol{v}}{\partial d_{n_d}^2}, \ldots\right) = 0, \quad (1)$$

where $\boldsymbol{\mathcal{F}}$ is a continuous function vector and n_d is the number of dimensions in the multi-dimensional space.

The procedure for solving the unknown function $\boldsymbol{v}$ in equation (1) is called the *solution to the PDE*. One of the numerical techniques used to approximate the solution to PDEs is the Finite Difference Method (FDM). FDM is based on approximating the PDEs through difference quotients and normally consists of three steps. First, an orthogonal grid or mesh where we want to find an approximate solution is generated. Then, the derivatives in a PDE or PDE system of equations are substituted with the finite difference schemes. Finally, the resulting linear/nonlinear system of algebraic equations is solved.

The target grid space has the same number of dimensions (n_d) as independent variables in the PDE. The local distance between the adjacent points in each dimension is defined as $\Delta d_1, \Delta d_2, \ldots, \Delta d_{n_d}$.

Finite difference schemes cover a wide array of solutions for PDEs. These solutions can be roughly divided into single-step and multi-step schemes. Single-step schemes approximates the solution by replacing the differential terms PDE with the corresponding finite difference terms. To illustrate single-step schemes, let us use the one-dimensional reaction-diffusion equation given by

$$\frac{\partial v}{\partial t} = D\frac{\partial^2 v}{\partial x^2} + f(v), \quad (2)$$

TABLE I

FINITE DIFFERENCE SCHEME EXAMPLES FOR THE ONE-DIMENSIONAL REACTION-DIFFUSION EQUATION.

(Scheme)	Discretization for 1D Reaction-Diffusion
FTCS	$\frac{v_{n+1,j} - v_{n,j}}{\Delta t} = D\frac{v_{n,j+1} - 2v_{n,j} + v_{n,j-1}}{(\Delta x)^2} + f(v_{n,j})$
BTCS	$\frac{v_{n+1,j} - v_{n,j}}{\Delta t} = D\frac{v_{n+1,j+1} - 2v_{n+1,j} + v_{n+1,j-1}}{(\Delta x)^2} + f(v_{n,j})$

where $f(v)$ is the reaction function, D is the diffusion coefficient and t and x are independent variables. v can be expressed as $v_{n,j}$, where the index terms n and j are used for the time and space dimension, respectively.

Applying the FTCS (Forward-Time Centered-Space) finite differencing scheme to equation (2) results to

$$\frac{v_{n+1,j} - v_{n,j}}{\Delta t} = D\frac{v_{n,j+1} - 2v_{n,j} + v_{n,j-1}}{(\Delta x)^2} + f(v_{n,j}). \quad (3)$$

The PDE discretization involves the replacement of the differential and arithmetic variables with their discrete counterpart given by

$$\boldsymbol{v} \rightarrow \boldsymbol{v}_{j_1,j_2,\ldots,j_{n_d}} \quad (4)$$

$$\boldsymbol{y} \rightarrow \boldsymbol{y}_{j_1,j_2,\ldots,j_{n_d}} \quad (5)$$

and of the differential operators, $\mathcal{L}_p^{(q)}$, with the specified PDE discretization scheme given by

$$\mathcal{L}_1^{(1)}(\boldsymbol{v}) \rightarrow \mathcal{L}_{\Delta 1,1}(\boldsymbol{v}), \quad (6)$$

$$\vdots$$

$$\mathcal{L}_{n_d}^{(m_{n_d})}(\boldsymbol{v}) \rightarrow \mathcal{L}_{\Delta n_d, m_{n_d}}(\boldsymbol{v}), \quad (7)$$

where $\mathcal{L}_{\Delta p,q}(\boldsymbol{v})$ corresponds to the discretized equation of the q^{th} derivative of $\boldsymbol{v}$ with respect to p. Aside from the explicit FTCS scheme in equation (3), other single-step finite difference schemes like the fully-implicit Backward-Time Centered-Space (BTCS) method (Table I) offer more numerical stability.

Our study will be limited to the code generation of biological simulations using explicit, single-step FDM. If an implicit scheme is used, it is necessary to use solvers for nonlinear systems of equations in the generated code and this is beyond the capability of our system as of the moment.

B. Simulation Model Discretization and Code Generation System Description

The main goal of our system is to automatically generate program codes for multidimensional simulations involving PDEs. This is an extension of the system we published earlier for generating biological simulation codes using ODE solving schemes [7]. In addition, the input files and algorithm include extensions from another study, which implements handling of multiphysics biological simulation [8].

The system is composed of three stages, namely, single-step PDE discretization, loop structure creation and program code generation. The stages are further discussed in the following subsections.

```
def method FTCSDiffusion1D as
    def variables pdesolvar as
        vartype diffvar: {$v$};
        vartype arithvar: {$y$};
        vartype constvar: {$z$};
        vartype dimensionvar: {$t, x$};
        vartype indexvar: {$n, j$};
        vartype deltavar: {$\Delta t$, $\Delta x$};
    enddef;
    $v \equiv v_{n,j}$;
    $y \equiv y_{n,j}$;
    $\frac{\partial v}{\partial t} \equiv \frac{v_{n+1,j} - v_{n,j}}{\Delta t}$;
    $\frac{\partial^2 v}{\partial x^2} \equiv \frac{v_{n,j+1} - 2v_{n,j} + v_{n,j-1}}{(\Delta x)^2}$;
enddef;
```

Fig. 1. The TecML information for the FTCS method in one dimension. The first part lists all the variables and their types while the second part enumerates the variable and operator discretization equations. The information is represented in COR notation [9].

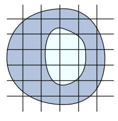
1) Single-Step PDE Discretization: The first stage in the system involves the discretization of the model variables, equations and boundary conditions with their corresponding discrete terms. Equations (4)–(7) represent the variable and differential operator discretization.

The inputs for the first stage are composed of a CellML or PHML file (cell model), a TecML file (differential solution scheme) and a RelML file (relation file). The TecML (Time Evolution Calculation Markup Language) file, introduced in our previous study [7] and describes ODE solution schemes, was extended in this study to describe the finite difference solutions for PDEs. We used RelML (Relation Markup Language) to describe the relationship between a CellML/PHML file and TecML file. The previous version of RelML maps the variables in the cell model file to their corresponding type in the numerical solution scheme file. This was expanded to include information about the morphology and boundary conditions for the underlying PDEs.

The single-step replacement of the model terms depends on the numerical scheme described in the TecML file. The TecML file lists the identifier for the predefined variable types, namely, differential, arithmetic and constants. It also contains the variables corresponding to each dimension as well as their respective indices and unit change per dimension. The information of a sample TecML file shown in Fig. 1 describes the FTCS scheme in one-dimensional space.

The system presents a general way of discretizing the PDEs by using the information provided with the geometry mesh points. The simulation mesh can be a 1D line, 2D rectilinear grid, or 3D cube. The geometry, boundary conditions, and distributed parameters are described as follows:

1) The geometry value is indicated for all nodes in the geometry mesh. The mesh node takes a value of 1 (*true*) value if the node contains a material (i.e., part of the biological simulation morphology) and a 0 (*false*) value if it is part of the background (i.e., empty space) (Fig.2(b)).
2) The boundary condition information, which contains either a Dirichlet or Neumann condition, are specified



(a) 2D morphology

$$\begin{matrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{matrix}$$

(b) morphology data

$$\begin{matrix} 0 & 5 & 1 & 6 & 0 \\ 7 & 3 & 0 & 2 & 9 \\ 2 & 3 & 0 & 0 & 3 \\ 2 & 3 & 0 & 0 & 3 \\ 8 & 3 & 0 & 7 & 10 \\ 0 & 11 & 4 & 12 & 0 \end{matrix}$$

(c) boundary condition id

Fig. 2. Morphology data (b) and the boundary condition id (c) specified in a RelML file corresponding to the morphology image information (a).

using unique boundary identification numbers. These numbers are used as reference to describe the boundary discretization or value for each bounded variable (Fig.2(c)).

3) In instances where distributed parameters are needed for the simulation, each distributed parameter has a designated file that contains the list of all the mesh nodes and the parameter value for each node.

The geometry, boundary condition, and distributed parameter mesh nodes and values are all written in individual CSV (comma-separated values) files. The file names are supplied in the RelML file to guide the system during the discretization process. Figure 3 shows the information in the RelML file, including the geometry, distributed parameter and boundary condition file name and identification numbers. These information are used to generate the discrete model equations for all instances of time and space to achieve generality in handling different numerical solution schemes and boundary conditions.

2) Loop Structure and Program Code Generation: Once all the discretized equations are generated, they are constructed into *do-while* loops by identifying repeating equations. Finally, the program code for C or C Cuda is generated from the loop structure [7]. The current implementation allows handling of implicit functions only in the code generation part.

III. Experimental Results and Analysis

Two sets of experiments were done to test the system. In the first experiment, a homogeneous and heterogeneous transmural sheet model of the cardiac muscle were used to show the distributed parameter model handling. The Luo-Rudy 1991 (LR1) model with the diffusion equation in two dimensions [10] was used for both models. The FTCS scheme was used to discretize the model equations. A 100×100 regular grid mesh was used in both simulations. In the heterogeneous grid, the value of the potassium channel conductance G_K varies for each section in the myocardial wall. The 15/100, 50/100 and 35/100 sections of the mesh represent the endocardial, middle and epicardial part of the cardiac wall, respectively. These sections has conductance values of 154%, 59% and 100 % of the original value from the LR1 model [11].

Figure 4 shows part of the expanded equations of the LR1 model equations for all instances of time and space. This results to a program code that has a triple-loop structure nested in three layers. The boundary conditions are also included inside the loops for the spatial dimensions. A section of the resulting program code generated by the proposed system is shown in Fig. 5.

```
def method LR1991_FTCSDiffusion2D as
    def cellml Luo-Rudy1991-2D as
        filename "Luo-Rudy1991-2D.cellml";
    enddef;
    def tecml FTCSDiffusion2D as
        filename "FTCSDiffusion2D.tecml";
    enddef;
    def variables pdesolvar as
        vartype diffvar: ${v, m, h, ...}$;
        vartype arithvar: ${I_{Na}, I_{si}, ...}$;
        vartype constvar: ${R, T, ...}$;
        vartype dimensionvar: ${t, x}$;
        vartype indexvar: ${n, j}$;
    enddef;
    def morphology 2D_WallModel as
        filename "geometry.csv";
    enddef;
    def boundarycondition 2D_WallModelBC as
        varname $v$;
        filename "boudarycondition.csv";
    enddef;
    def parametercondition 2D_WallModelGks as
        varname $G_{Ks}$;
        filename "Gks.csv";
    enddef;
    def boundary-condition-id "1" as
        $\frac{\partial v}{\partial x} = 0$;
    enddef;
enddef;
```

Fig. 3. Information contained in the RelML file for 2D Luo-Rudy 1991 excitation propagation model combined with the FTCS scheme. The first part describes the relation between the variables in the two files. The spatial geometry, spatial boundary condition, and the actual boundary condition information are also included.

```
/***** nondifferential equations *****/
$\alpha_m[1][1][1] = 0.32 * (V_m[1][1][1] + 47.13)/(1 - exp(\cdots$
$\alpha_h[1][1][1] = (V_m[1][1][1] < -40)?0.135 * exp(\cdots$
⋮
/***** differential equations *****/
$V_m[1+1][1][1] = V_m[1][1][1] + (-1/C[1][1][1]) * (I_{stim} \cdots$
$m[1+1][1][1] = m[1][1][1] + \alpha_m[1][1][1] * (1 - m \cdots$
⋮
```

Fig. 4. Equations for the mesh point (1, 1, 1) that are generated by first step of the proposed system by using the FTCS scheme on the Luo-Rudy 1991 2D excitation propagation model.

The stimulation current program code was manually inserted in this experiment in order to create the formation of a spiral wave during simulation. Note that this can be represented as a stimulation current equation with a distributed parameter. The difference between the homogeneous and distributed parameter mesh was observed with the former allowing for the formation of a spiral wave (Fig. 6).

In the second experiment, the same equations and parameters were applied to different morphology models. A simple ring shape and a middle heart cross-section slice shape were used to show the handling of complex morphology, boundary condition and distributed parameter data. The sample simu-

```
n = 0;
do {
  j = 1;
  do {
    l = 1;
    do {
      if (j==1) V_m[n+1][0][l]=0;
      :
      V_m[n+1][j][l] = V_m[n][j][l] + ···
      :
      l = l+1;
    } while (l<=100);
    j = j+1;
  } while (j<=100);
  n = n+1;
} while (n<=399);
```

Fig. 5. The resulting program code generated by the second step of the proposed system for the combination of Luo-Rudy 1991 2D excitation propagation model and FTCS scheme. In this example, all the unknown variables can be calculated by explicit calculation, thus the sequential calculation program code is generated.

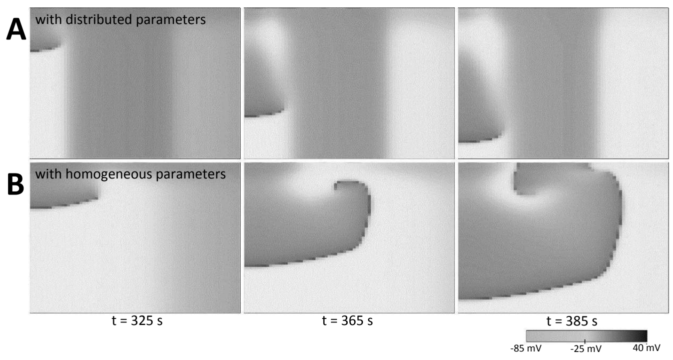


Fig. 6. The action potential propagation simulation of the Luo-Rudy 1991 model in a sheet with distributed (A) and homogeneous (B) parameters for the potassium channel conductance G_K. Due to the differences in the action potential duration (APD) of the sheet with distributed parameters, a spiral wave did not form during simulation. However, a spiral formed for the sheet with homogeneous parameters.

lation results and excitation patterns are shown in Fig. 7.

The system poses some limitations on the generation of program codes in terms of handling a large number of equations. For the Luo-Rudy model simulation, there are a total of eight differential and 31 arithmetic equations. It also includes 12 boundary instances from four boundary condition equations in a 2D mesh. For a simulation with 100×100 mesh size and 400 time steps, this means a total of at least 172 million equations that need to undergo dependency analysis. Since it is not feasible to handle such number of equations, the system uses a sample size in each dimension to determine the variable dependencies instead. In addition, efficient program codes can be generated for implicit equations only if implicit solutions are needed in all the indices or dimensions.

IV. CONCLUSIONS

In this study, we proposed a general way of applying finite difference schemes to partial differential equations on uniform rectilinear grids. Our system handles morphology

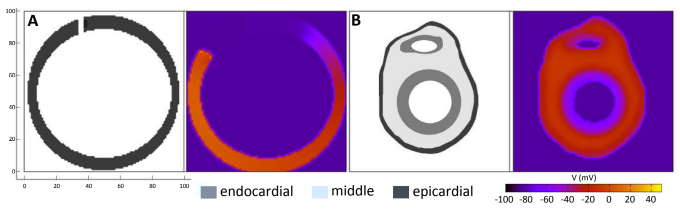


Fig. 7. The cardiac morphology and respective sample result for a ring-shaped tissue (A) and middle heart cross-section (B) simulation of the Luo-Rudy 1991 model. A mesh size of 100×100 was used for the simulations. The mesh was divided into the endocardial, mid and epicardial sections for the middle heart cross-section morphology.

and boundary condition information automatically using a markup language file. Through the proposed method, it is possible to automatically generate simulation program codes for physical and biological functions modeled by distributed parameter systems.

For further studies, we would like to support other discretization methods like finite volume (FVM) or finite element (FEM) method to handle more complex geometries. FDM was used for its simplicity and ease of implementation but it is not suited for complex geometries. FVM and FEM will also allow us to handle unequal grid spacing. In addition, the system also needs to include handling of implicit schemes and code generation of nonlinear systems of equation solvers.

REFERENCES

[1] P. Hunter, P. Robbins and D. Noble. "The IUPS human physiome project." *Eur J Physiol*, vol. 445(1), pp. 1–9, 2002.

[2] M. Hucka, H. Kitano et al. "The systems biology markup language (sbml): A medium for representation and exchange of biochemical network models." *Bioinformatics*, vol. 19(4), pp. 524–531, 2003.

[3] A.A. Cuellar, C.M. Lloyd, P.F. Nielsen, D.P. Bullivant, D.P. Nickerson and P.J. Hunter. "An overview of cellml 1.1, a biological model description language." *SIMULATION*, vol. 79(12), pp. 740–747, 2003.

[4] Y. Asai, Y. Suzuki, Y. Kido, H. Oka, E. Heien, M. Nakanishi, T. Urai, K. Hagihara, Y. Kurachi and T. Nomura. "Specifications of insilicoML 1.0: A multilevel biophysical model description language." *The Journal of Physiological Sciences*, vol. 58(7), pp. 447–458, 2008.

[5] G. Christie, P. Nielsen, S. Blackett, C. Bradley and P. Hunter. "FieldML: concepts and implementation." *Phil. Trans. R. Soc. A*, vol. 367, pp. 1869-1884, 2009.

[6] D. Chang, N. Lovell and S. Dokos. "Field Markup Language: Biological field representation in XML." in *Proc. IEEE EMB Conf.*, 2007, pp. 402-405.

[7] F.R. Punzalan, Y. Yamashita, N. Soejima, M. Kawabata, T. Shimayoshi, H. Kuwabara, Y. Kunieda and A. Amano. "A CellML Simulation Compiler and Code Generator using ODE Solving Schemes Description." *Source Code for Biology and Medicine*, vol.7(1):11, 2012.

[8] M. Kawabata, Y. Yamashita, F.R. Punzalan, Y. Kunieda and A. Amano. "A program code generator for multiphysics biological simulation using markup languages." in *Proc. Int. Workshop Innovative Architecture for Future Generation High-Performance Processors and Systems*, 2012.

[9] A. Garny, D. Nickerson, J. Cooper, R. Santos, A. Miller, S. McKeever, P. Nielsen and P. Hunter. "CellML and associated tools and techniques." *Phil. Trans. R. Soc. A*, vol. 366(1878), pp.3017-3043, 2008.

[10] R.B. Huffaker, J.N. Weiss, B. Kogan. "Effects of early afterdepolarizations on reentry in cardiac tissue: a simulation study." *Am J Physiol Heart Circ Physiol.*, vol. 292(6), H3089-102, 2007.

[11] T. Suzuki, M. Inagaki, T. Yao, T. Namba, R. Suzuki, M. Sugimachi, K. Nakazawa. "How M cell affects the polarity of T waves: a computer simulation of the transmural distribution of APD in the ventricle wall." *Technical Report of IEICE, MBE* vol. 101(406), pp. 21-28, 2001.