Adaptive grid algorithm in application to parallel computation of the solution of reaction diffusion equations

Joseph Yeh, Ray Huffaker, Dr. Boris Kogan
UCLA Computer Science Dept.

Abstract

The power of modern computers becomes insufficient for simulation of excitation wave propagation in two- and three-dimensional cardiac tissues composed of up-to-date mathematical cell models. Recently, a new idea has been proposed based on replacing the grid of constant space steps by a grid with space steps adapted to chosen characteristics of the wave processes. This approach has only been applied to problems with simplified cardiac cell models (Fitzhugh-Nagumo, Luo-Rudy I) on sequential computers. Here we present preliminary results of modification of this approach to study excitation wave propagation in cardiac tissue, with mathematical cell models containing intracellular Ca^{2+} dynamics, using parallel supercomputers. Using the operator splitting algorithm, the adaptive grid was implemented in solving the point model equations, while diffusion was solved using the original constant space step grid. Simulation results showed savings of up to 25% of program execution time for 1D simulation, and up to 52% of program execution time for 2D simulation. However, wave conduction velocity increased as the threshold for refining the adaptive grid was increased.

Introduction

The power of modern computers becomes insufficient for simulation of excitation wave propagation in two- and three-dimensional cardiac tissues composed of up-to-date mathematical cell models [1,2,3]. Proposed solutions to this problem have included: the operator splitting algorithm [4], introduction of the variable time step depending on the maximum rate of change of problem variables [5], tabular representation of non-linear functions, the use of parallel supercomputers, and optimal apportioning of these parallel processors among the nodes of a given grid. Recently, a new idea has been proposed based on replacing the grid of constant space steps by a grid with space steps adapted to chosen characteristics of the wave processes [6,7]. In the original paper this approach was used to study a simplified type of cardiac reaction-diffusion equations on sequential computers. Here we present preliminary results of modification of this approach to study excitation wave propagation in cardiac tissue, with mathematical cell models containing intracellular Ca^{2+} dynamics, using parallel supercomputers.

Methods

Constant Space Step Methods:

The mathematical model of wave propagation in 2D tissue is described by the following partial differential equation:

$$\frac{\partial V}{\partial t} = D_x \frac{\partial^2 V}{\partial x^2} + D_y \frac{\partial^2 V}{\partial y^2} + (I_m + I_{st}(s,t)) \frac{1}{C_m} \qquad (1)$$

with appropriate initial and boundary conditions.

Here V is membrane potential, D_x and D_y are diffusion coefficients, I_{st} is the external stimulating current, I_m is the net membrane ionic current, and C_m is a membrane capacity. To make the above equation closed, it is necessary to add the system of nonlinear ordinary differential equations (ODEs) that describes the behavior of all components of I_m and processes in intracellular compartments. For this purpose we have chosen the AP model proposed by Luo-Rudy [1] and modified by Chudin, et al. [8].

Computer simulations were performed on an IBM SP RS/6000 massively parallel computer at Lawrence Berkeley National Laboratory. Information was passed between processors using Message Passing Interface (MPI). The simulations used the operator splitting algorithm [4]. According to this algorithm the integration (1) is split into two parts: integration of diffusion equation $\frac{\partial V}{\partial t} = D_x \frac{\partial^2 V}{\partial x^2} + D_y \frac{\partial^2 V}{\partial y^2}$, and integration of the system of ODEs $\frac{\partial V}{\partial t} = (I_m + I_{st})\frac{1}{C_m}$. These integrations were executed in consecutive computational cycles of predetermined duration $\Delta t = 0.1$ ms. The specific features of this algorithm are described in Kogan, et al. [5] and Chudin, et al. [9]. The operator splitting algorithm allows integration of the system of nonlinear ODEs at any point in space independently and with variable time steps.

One computational cycle, $\Delta t = 0.1$ ms, is a three-step process: (1) Solving the diffusion equation for $\Delta t = 0.05$ ms, with the initial condition for ***V(x,y)*** taken from the end of the previous computational cycle; (2) Solving the system of ODEs for $\Delta t = 0.1$ ms, using variable time step from 0.005-0.1 ms, with the initial conditions for ***V(x,y)*** taken from step 1 and for gate variable values taken from step 2 of the previous computational cycle; (3) Solving the diffusion equation for $\Delta t = 0.05$ ms, with the initial condition for ***V(x,y)*** taken from step 2. This process is illustrated in Figure 1.

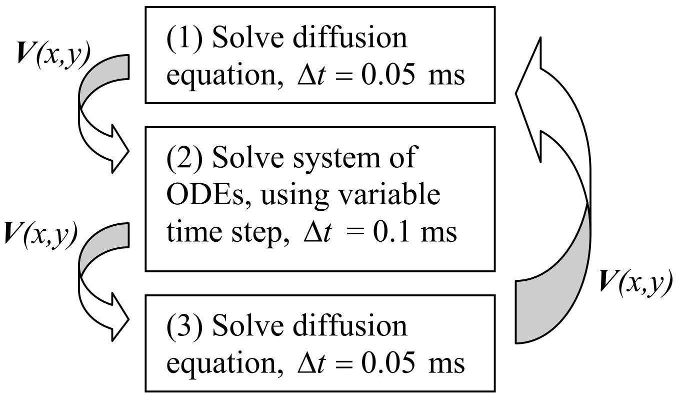
V(x,y)
(1) Solve diffusion equation, Δt = 0.05 ms
(2) Solve system of ODEs, using variable time step, Δt = 0.1 ms
V(x,y)
(3) Solve diffusion equation, Δt = 0.05 ms
V(x,y)

Figure 1: One computational cycle, $\Delta t = 0.1$ ms, and flow of V data

In the case of a constant space step, Δx was fixed at 0.025 cm for 1D and 2D simulations. Along with the variable time step of 0.005-0.1 ms, these choices provided stability and accuracy with the chosen numerical integration method, and did not disturb the conditions of medium continuity proposed by Winfree [10]. This condition requires that the chosen space step Δx satisfies $D > \Delta x^2 / T_r$. Here T_r is an activation rising time measured in a cell placed in tissue. This time is longer than in an isolated cell due to the effect of the local currents and incomplete recovery of I_{Na} during wave circulation with comparatively short period. Estimating $T_r = 2.5$ ms, we found that for $D = 1$ cm^2/s the inequality is satisfied for $\Delta x < 0.005$ cm.

An explicit Euler numerical method was used to solve the diffusion equation. The problem of computational error arises for solutions to the system of nonlinear ODEs describing the fast membrane processes during depolarization phase of the Action Potential (AP). To decrease these errors, we used the Euler explicit method with variable time steps, which were changed depending on the rate of the most rapid variable. In addition, for the fast sodium channel gate variable, we replaced the Euler method with the so-called hybrid method [15].

Modifications for Adaptive Grid:

The adaptive grid algorithm introduces a variable space step for integration of the system of ODEs. While the underlying (or "finest") grid remains the same (no nodes are added or taken away), integration of the ODEs for each computational cycle is only performed on a selected "adaptive grid" of nodes. Inclusion of nodes on the adaptive grid during a particular computational cycle depends on the distribution of dV/dx through the grid. In areas with large dV/dx, such as near the front of a propagating wave, the density of the adaptive grid must be larger than in areas of small dV/dx. Since the ODEs do not need to be integrated for nodes not selected for the adaptive grid during a particular computational cycle, the adaptive grid algorithm offered potential time savings for simulations.

One potential problem with such an algorithm occurs when a node that has not been on the adaptive grid for a few computational cycles is included in the adaptive grid at a later time. What initial values for state variables and V should be used to solve the ODEs? Normally we would use the output from the previous computational cycle, but in this case there was no output since the calculations were not done for this node in the previous cycle. The proposed solution to this problem is to estimate the values by interpolating from the values of the variables at surrounding cells that were included on the adaptive grid in the previous computational cycle. Using interpolation to estimate V after every ODE integration step gives us a value for V at every node, even those where the ODE integration was not performed. Thus, the diffusion equation can be integrated using the finest grid. One computational cycle in the adaptive grid algorithm now requires that step 2 from before be split into three parts: (2a) Select the adaptive grid, based on distribution of V; (2b) Solve system of ODEs for adaptive grid points; (2c) Interpolate values of V and other state variables for non-selected nodes. The new process can be seen in Figure 2.

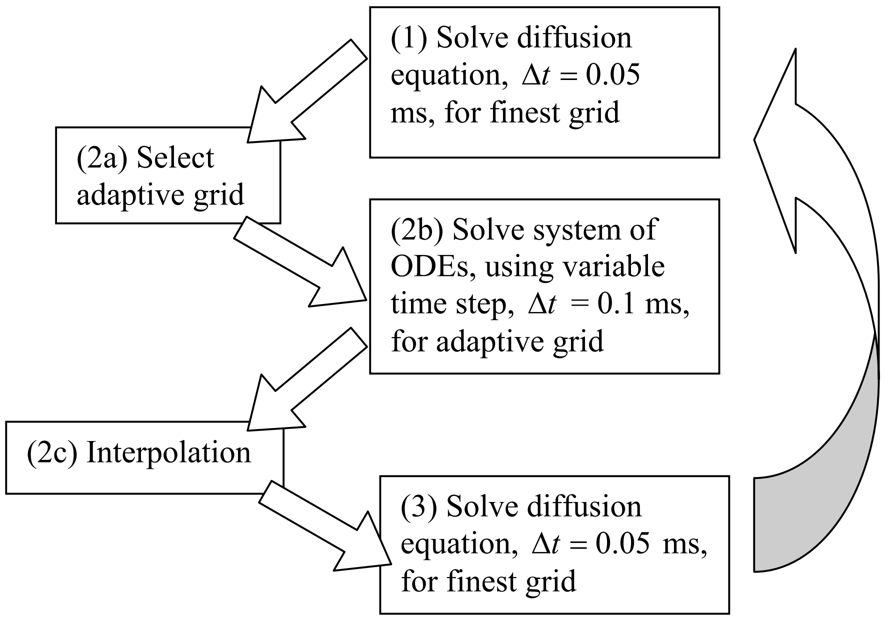
(1) Solve diffusion equation, $\Delta t = 0.05$ ms, for finest grid
(2a) Select adaptive grid
(2b) Solve system of ODEs, using variable time step, $\Delta t = 0.1$ ms, for adaptive grid
(2c) Interpolation
(3) Solve diffusion equation, $\Delta t = 0.05$ ms, for finest grid

Figure 2: One computational cycle for adaptive grid algorithm, $\Delta t = 0.1$ ms

The basis for the adaptive grid is the "coarsest grid", a set of nodes that will be automatically selected for the adaptive grid during every computational cycle. We choose a coarsest grid composed of every fourth node, yielding a maximum space step of $4^* \Delta x$ for the grid (see Figure 3). We will define a basic grid unit (BGU) as the smallest subsection of the coarsest grid. For 1D, a BGU would be a 5 x 1 node piece of the grid, with coarse nodes at either end. For 2D, a BGU would be a 5 x 5 node piece of the grid, with coarse grid points at every corner. An n x n node 2D grid would then be composed of (n/4)*(n/4) BGUs.

As tissue size grows much larger than the size of an individual BGU, the coarsest grid offers a potential reduction of ODE calculations approaching 75% (only necessary to calculate 1 out of every 4 nodes) in the case of 1D and approaching 93.75% (calculate only 1 out of every 16 nodes) in the case of 2D.

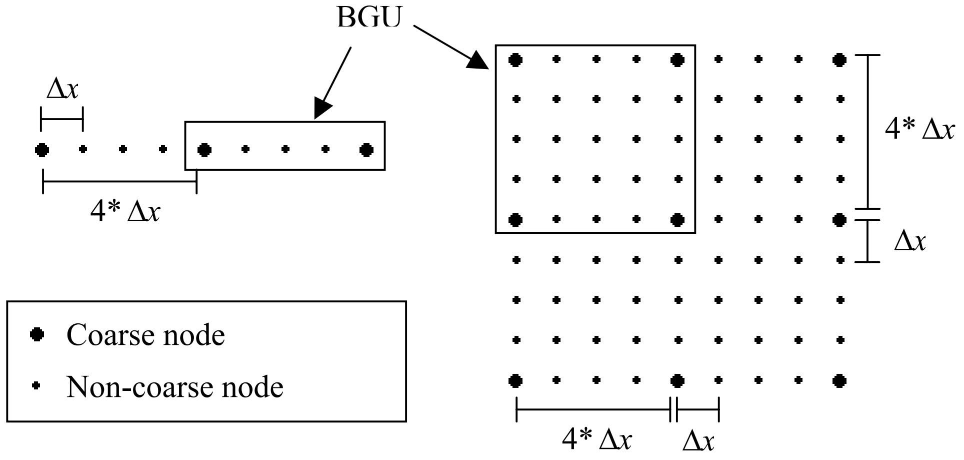

Figure 3: Coarsest grid (represented by large dots) and BGU (outlined) for 1D (9 x 1 node grid) and 2D (9 x 9 node grid) cases

In every computational cycle, the adaptive grid will include all the coarse grid points. The other nodes selected for the adaptive grid are chosen according to the distribution of V across the grid outputted from the previous diffusion calculation. The first step is choosing the threshold ΔV so that the grid will be refined in areas where voltage varies beyond ΔV. Larger values of ΔV should decrease program execution time by omitting more ODE integrations, but might also decrease the accuracy of the results. Our simulations were run with ΔV values varying from 0 to 5 mV.

Selection of nodes for the adaptive grid is performed on each BGU. For the 1D case, nodes are selected based on a recursive method. Define V_i as the voltage of the node at position *i*. If, for the two coarse grid points X_i and X_{i+4} of the BGU, $|V_{i+4} - V_i| \geq \Delta V$, then` node X_{i+2} must be added to the adaptive grid. If X_{i+2} is indeed selected for the grid, we recursively examine $|V_{i+2} - V_i|$ and $|V_{i+4} - V_{i+2}|$, adding to the adaptive grid X_{i+1} and X_{i+3}, respectively, if the differences exceed ΔV.

Selection of nodes for the 2D case is similar. The nodes of a BGU are given a hierarchy, seen in Figure 4. Each node will have two or four nodes of a higher level, which we will call critical nodes, that determine whether or not they will be added to the grid. Level 1 consists of the coarse grid points at each corner of the BGU. These are automatically selected for the adaptive grid. Level 2 consists of the four nodes in the middle of any two level 1 nodes. The critical nodes for each level 2 node are the two level 1 nodes it is between. If V varies beyond ΔV for any two level 1 nodes, then the level 2 node between them is added. For example, if $|V_{i+4,j} - V_{i,j}| \geq \Delta V$, then node $X_{i+2,j}$ is added to the adaptive grid. Level 3 consists only of the node in the middle of the BGU. Its critical nodes are the four level 2 nodes. If more than one of the level 2 nodes were added to the adaptive grid, then the level 3 node is added as well. Level 4 consists of the twelve nodes that are bordered either horizontally or vertically by two nodes of level 1, 2, or 3. The two higher-level nodes bordering the level 4 node are its critical nodes. If both were added to the adaptive grid, and they have V variance beyond ΔV, then the level 4 node is added to the adaptive grid. For example, if $X_{i+2,j+2}$ and $X_{i+2,j+4}$ were both added to the adaptive grid, and $\left|V_{i+2,j+2} - V_{i+2,j+4}\right| \geq \Delta V$, then node $X_{i+2,j+3}$ is added to the adaptive grid. Level 5 consists of the four remaining nodes, each of which is bordered by four level 4 nodes, its critical nodes. A level 5 node is added to the adaptive grid if more than one of its bordering level 4 nodes was added to the grid.

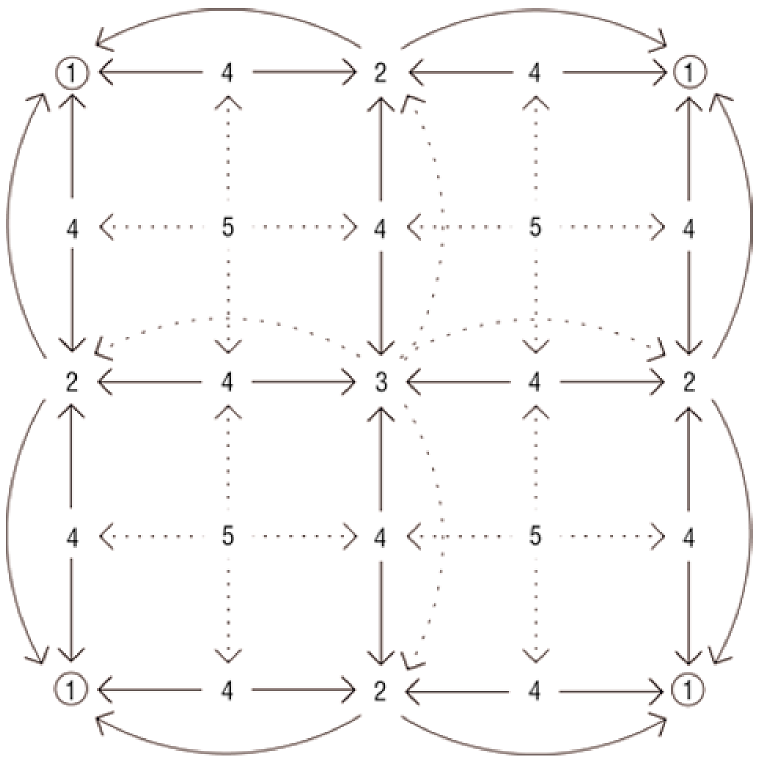

Figure 4: Hierarchy of nodes in a 2D BGU. Arrows point from each node to its critical nodes. Solid arrows indicate node is added to the adaptive grid if V difference between its critical nodes exceeds ΔV. Dotted arrows indicate node is added to the adaptive grid if more than one of its critical nodes were added.

Once the adaptive grid has been selected for a computational cycle, the system of ODEs is integrated only in the selected nodes. Following integration, V and gate variable values in non-selected nodes must be interpolated from the calculated values in selected nodes. Like grid selection, interpolation is performed on each BGU. In the 1D case, interpolation is again performed recursively. If X_{i+2} was added to the adaptive grid, then we performed the calculations there and do not need to interpolate for that node. If X_{i+2} was not added to the adaptive grid, then we set $V_{i+2} = \frac{V_i + V_{i+4}}{2}$. For nodes X_{i+1} and X_{i+3}, if they were not added to the adaptive grid then we set $V_{i+1} = \frac{V_i + V_{i+2}}{2}$ and $V_{i+3} = \frac{V_{i+2} + V_{i+4}}{2}$. If X_{i+2} was not added to the adaptive grid, then we will be finding interpolated values of V_{i+1} and V_{i+3} using an interpolated value of V_{i+2}. All other state variables are interpolated in the exact same manner as V.

Interpolation for the 2D case is done using the same hierarchy of nodes as adaptive grid selection. For every node that was not selected for the adaptive grid, V is set to the average V of its critical nodes. All other state variables for non-selected nodes are also set to the average of the state variables in the critical nodes.

Applying the adaptive grid algorithm to the setting of parallel computation requires some special consideration for the interpolation algorithm. Two possible methods of interpolation are possible: inter-processor and intra-processor. In the inter-processor method, interpolation would be done assuming knowledge of the entire grid. It would be performed sequentially using MPI to gather data from all processors. The intra-processor method would entail each processor independently interpolating upon its own pre-designated subsection of the grid. No MPI would be required during the interpolation since each processor would only need to know the information it already has. In order for the intra-processor method to work, the perimeter of each processor's independent subsection of the grid would need to be included on the adaptive grid at all times, i.e. added to the coarsest grid, since the perimeter nodes would have no knowledge of their neighbors (see Figure 5). Thus, the inter-processor method will be slowed by MPI communication while the intra-processor method will be slowed by more nodes needing to be on the adaptive grid.

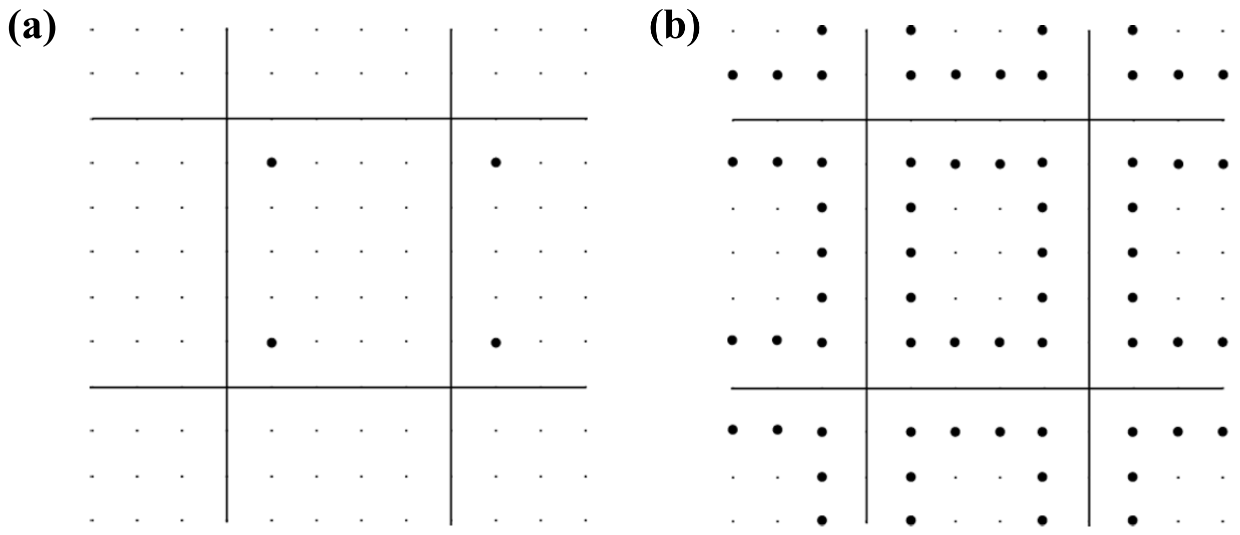

Figure 5: Coarsest grid for portions of a 2D grid using the (a) inter- and (b) intra-processor interpolation algorithms. Lines indicate division of the grid between processors.

Distribution of the grid across processors for the 2D case will be affected by which interpolation method is used. The inter-processor method would favor a distribution of the grid with a high perimeter/area ratio for the subsections of the grid given to each processor, since time needed for MPI communication depends more on the total number of communications between processors, rather than the size of each communication. The intra-processor method would favor distribution of the grid with a low perimeter/area ratio since this would decrease the number of nodes on the perimeter, all of which need to be added to the coarsest grid.

Since the adaptive grid introduces a variable space step, we must reconsider whether the conditions of stability and continuity are still met. The stability condition is $\Delta t \leq \frac{\Delta x^2}{2D}$ for 1D and $\Delta t \leq \frac{\Delta x^2}{4D}$ for 2D. This condition is satisfied by the adaptive grid method, since increasing the space step loosens the bound on Δt. However, recall the continuity condition $D > \Delta x^2 / T_r$, requiring in our problem that $\Delta x < 0.05$ cm. This is true only when the adaptive grid reaches the highest level of refinement. Thus, we must ensure that we choose an adequate threshold ΔV such that near the front of a propagating wave, where the continuity condition applies, the adaptive grid contains all the nodes.

Results

All simulations were run for 500 ms. For 1D simulations, the first 20 nodes of the grid were given a stimulus current to induce a propagating wave in the tissue. Simulation of the adaptive grid algorithm with inter-processor interpolation for a 1D tissue yielded bad results (see Table 1). The program executed more than 7x slower than the original program with constant Δx. Message passing amongst processors proved to be an immensely time consuming solution. Switching between parallel and sequential processing consumed time as all processors needed to finish running before interpolation could be done. The slowest processor became the rate-limiting factor and hindered the progress of the entire program.

Table 1:

1D, inter-processor:	# of Processors:	Grid Size (# of nodes)	Simulation Time	ΔV (mV)	Program Execution Time
Constant Δx	64	4096×1	500 ms	n/a	24.246 s
Adaptive grid	64	4096×1	500 ms	1	172.804 s

The adaptive grid algorithm with intra-processor interpolation in a 1D tissue yielded better results (see Table 2). The program executed faster than in the original program with constant Δx. 21% of the execution time was saved with a threshold value of $\Delta V = 1$ mV, while 25% of the time was saved with $\Delta V = 5$ mV.

Table 2:

1D, intra-processor	# of Processors:	Grid Size (# of nodes)	Simulation Time	ΔV (mV)	Program Execution Time
Constant Δx	64	2112×1	500 ms	n/a	17.822 s
Adaptive grid	64	2112×1	500 ms	0	14.983 s
Adaptive grid	64	2112×1	500 ms	1	14.106 s
Adaptive grid	64	2112×1	500 ms	3	13.732 s
Adaptive grid	64	2112×1	500 ms	5	13.325 s

Visualization of the voltage output from the adaptive grid at different times showed the coarsest grid being used in areas away from the wave front, and a finer grid being used near the wave front (see Figure 6). As the threshold ΔV increased, the area of the fine grid used near the front of the wave becomes smaller and coarser.

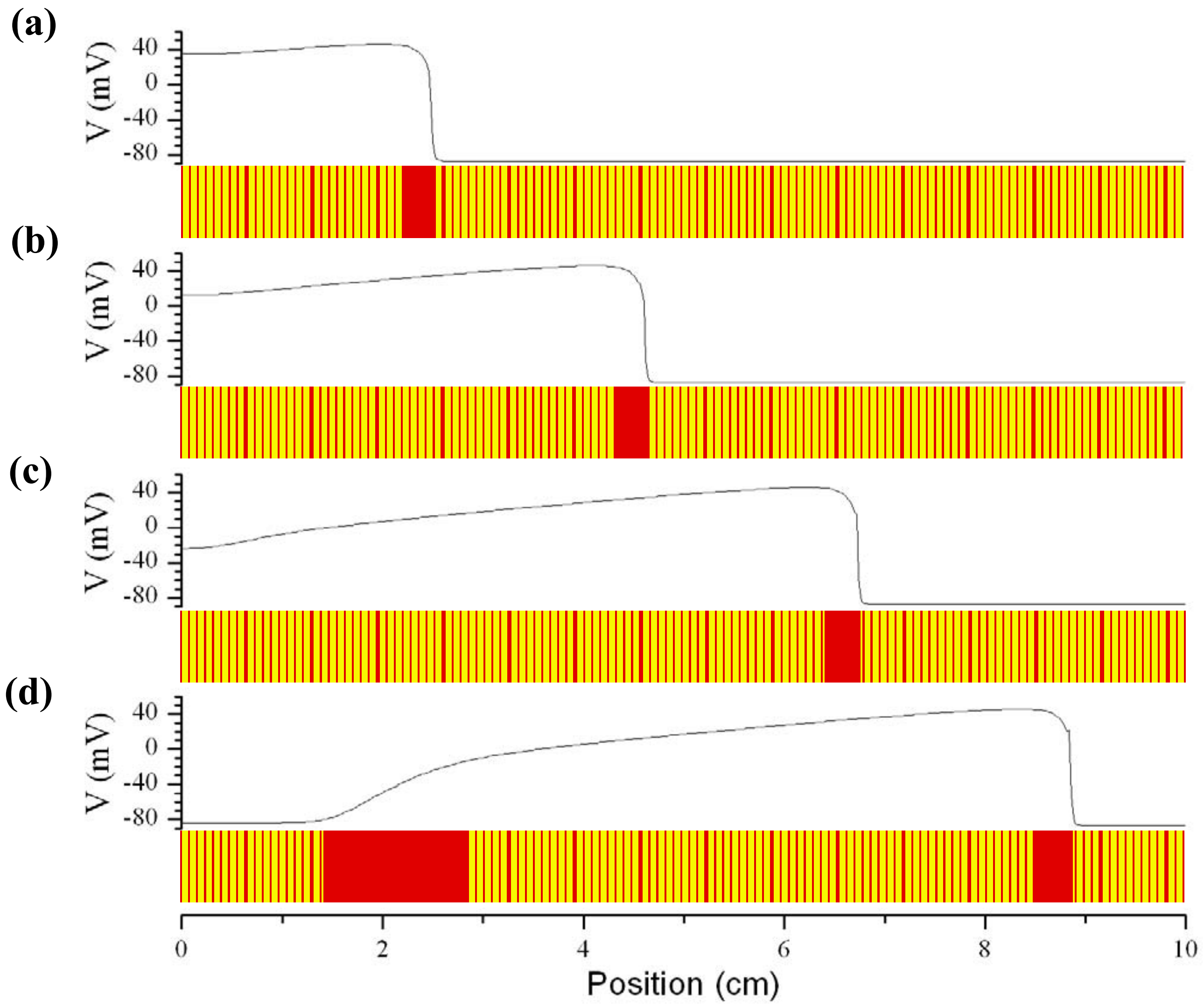

Figure 6: Voltage trace and visualization of adaptive grid with $\Delta V = 1$ mV at times: (a) 50 ms, (b) 100 ms, (c) 150 ms, (d) 200 ms. Adaptive grid is shown below corresponding voltage trace. Black vertical lines indicate that node has been selected for the adaptive grid, while gray areas indicate regions of nodes not selected for the adaptive grid. Area of fine grid near wave tail in (d) is due to rapid repolarization after plateau.

Visualization of also showed an unwanted side-effect of the adaptive grid algorithm. For $\Delta V = 0$ mV, the conduction velocity of the wave (θ) was the same as in the constant space step case. However, as the ΔV increased, θ increased (see Figure 7a). Subsequently, the wave propagated further along the tissue during the 500 ms simulation as ΔV increased (see Figure 7b).

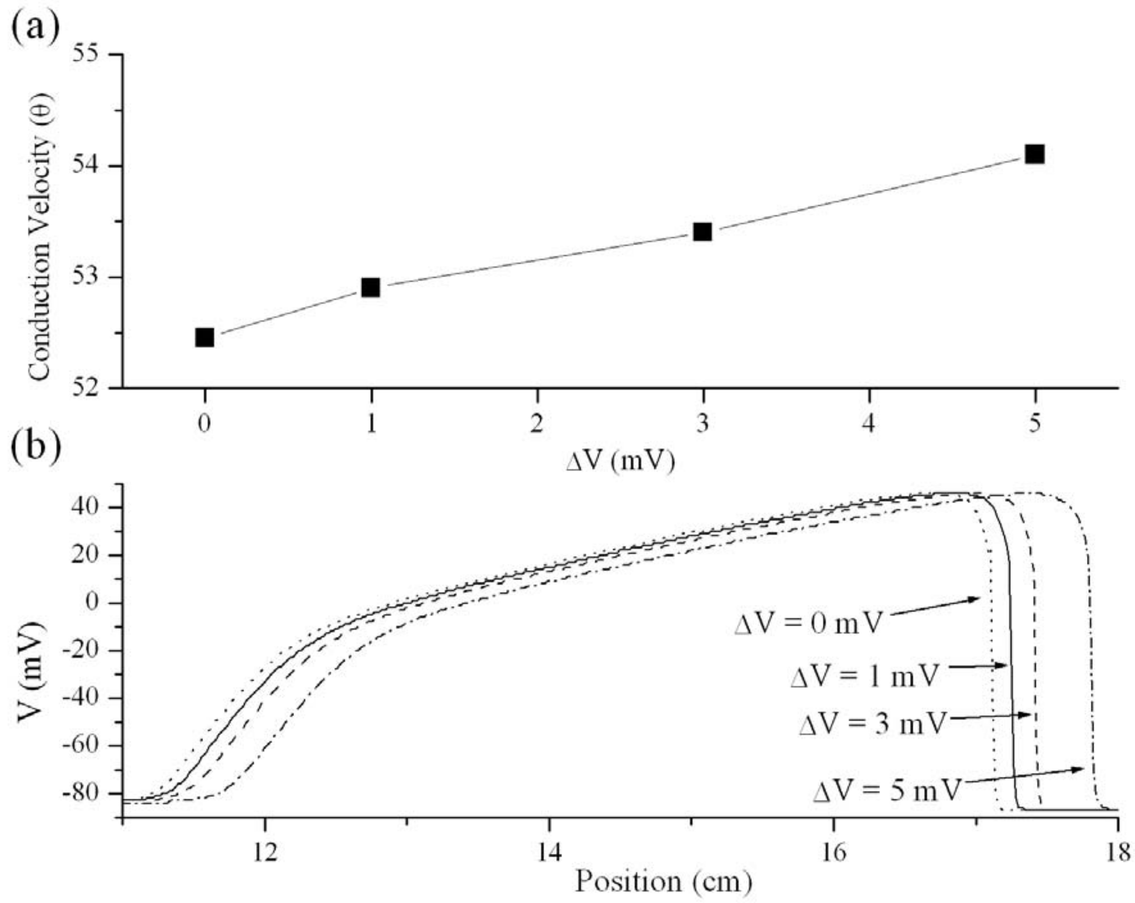

Figure 7: 1D simulations of adaptive grid algorithm for different values of ΔV. (a) Conduction velocity (θ) increases as ΔV increases. (b) Position in tissue of wave front after 500 ms. Wave has propagated further in the case of larger ΔV.

For 2D simulation a stimulus current was given in the first 20 columns of nodes in the tissue to induce rectilinear wave propagation. A premature beat was given near the tail of the propagating wave to induce a spiral wave. In all simulations the time and location of the premature beat was the same. The premature beat was given at t = 238 ms, meaning the 500 ms simulations had 238 ms of rectilinear wave propagation and 262 ms of spiral wave formation and rotation. Rectilinear wave propagation should be less computationally intensive than the spiral wave formation and rotation since there is less variation of voltage among nodes over the entire grid during rectilinear propagation.

The adaptive grid algorithm showed greater time savings than in the case of 1D simulation (see Table 3). With ΔV = 0 mV the program ran slower than the original. However, 22% of the execution time was saved for ΔV = 1 mV and 52% of the execution time was saved for ΔV = 5 mV.

Table 3:

2D, intra-processor	# of Processors:	Grid Size (# of nodes)	Simulation Time	ΔV (mV)	Program Execution Time
Constant Δx	64	264×264	500 ms	n/a	364.653 s
Adaptive grid	64	264×264	500 ms	0	382.788 s
Adaptive grid	64	264×264	500 ms	1	283.952 s
Adaptive grid	64	264×264	500 ms	3	244.687 s
Adaptive grid	64	264×264	500 ms	5	173.312 s

Visualization of the voltage output for the 2D case showed the same unwanted side-effect as in 1D (see Figure 8). For $\Delta V = 0$ mV, the wave propagated and formed a spiral wave in the same manner as the original program. However, for $\Delta V = 1$ mV and $\Delta V = 3$ mV we see a slight increase of rectilinear conduction velocity. The subsequent spiral wave then propagates slower, due to an increased refractory period in the region of the premature beat. When $\Delta V = 5$ mV, the rectilinear wave propagates so quickly that the premature beat arrives much too late and forms a circular wave rather than a spiral wave.

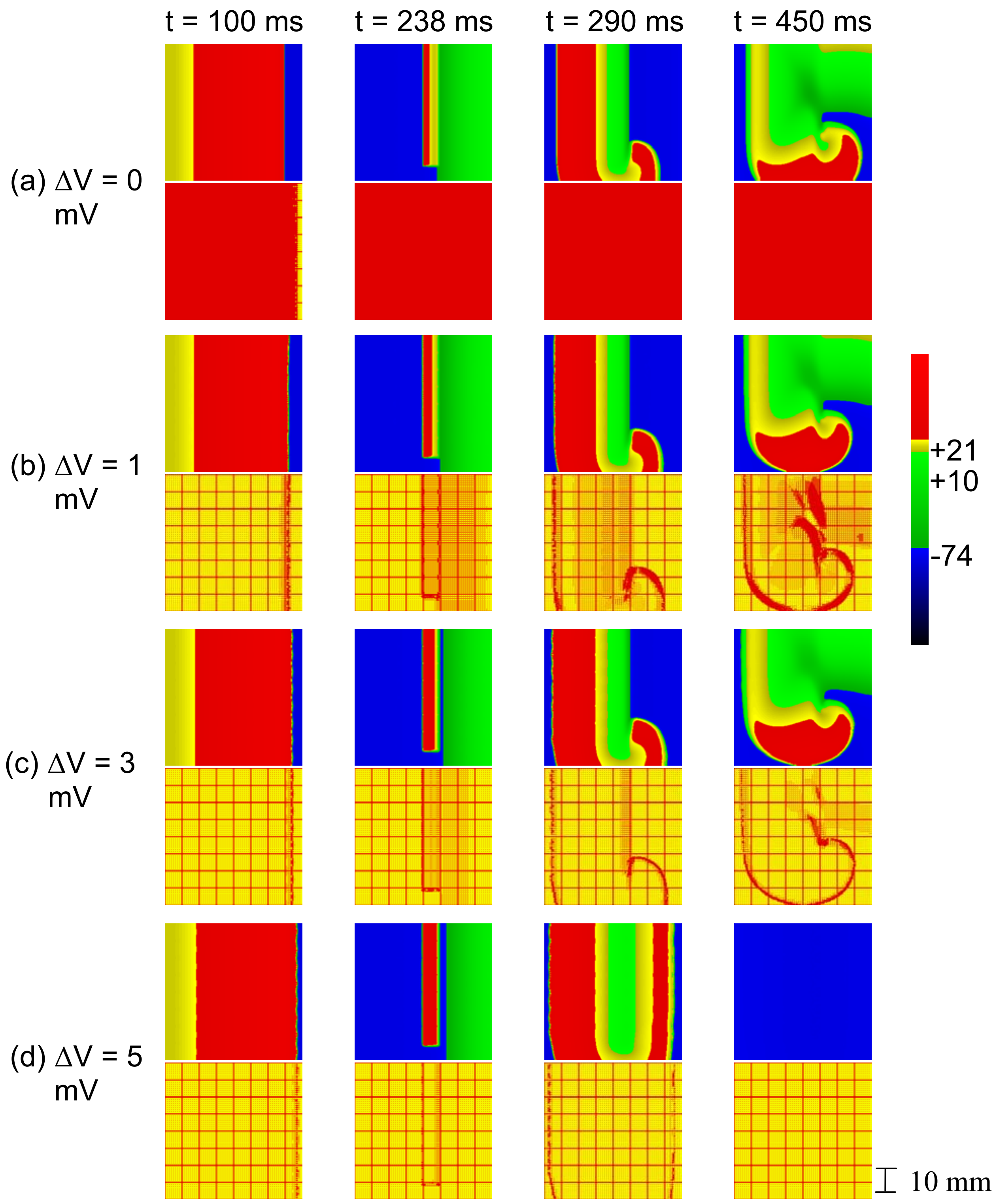

Figure 8: Voltage traces and visualization of adaptive grid at various times for 2D simulation and varying ΔV. Rectilinear wave propagation (t = 100 ms), premature beat (t = 238 ms), formation of the spiral wave (t = 290 ms), and first turn of the spiral wave (t = 450 ms) are shown. Adaptive grid is shown below corresponding voltage trace. Black dots indicate that node has been selected for the adaptive grid, while gray areas indicate regions of nodes not selected for the adaptive grid.

Conclusion

These preliminary results with the adaptive grid method indicate it may be a very useful device for the problem of simulation of excitation wave propagation in cardiac tissue. The algorithm cut program execution time by about ¼ in the 1D case and about ½ in the 2D case. Time savings in 3D tissue could be even greater, because the coarse grid points will make up less of the total grid (1/64 of the grid in 3D, compared to 1/16 in 2D and ¼ in 1D). However, the issue of intra-processor interpolation looms large in 3D. Clearly, the MPI time needed for inter-processor interpolation is unrealistic. Simulations using inter-processor interpolation took 7x longer than the constant space step program. Thus, intra-processor interpolation must be used. With intra-processor interpolation, though, every node on the border of a processor's subsection of the grid needs to be added to the adaptive grid. In 1D the border between processors is two nodes, in 2D it is the perimeter of a rectangle, and in 3D it will be the entire surface of a rectangular prism. Reconfiguring the method of intra-processor interpolation may be necessary.

Another problem to be dealt with is the observed increase of speed of propagation in 2D tissue by the adaptive grid method. We suspect that this is the result of the interpolation algorithm. State variables are interpolated across the tissue. However, only V changes in space. The other state variables change due to intracellular processes and do not diffuse. Instead of using interpolation, these values could be calculated from the differential equations using the value from the last time the node was on the adaptive grid. Another alternative that could potentially solve this problem is to extrapolate rather than interpolate. Many of the state variables, such as the gate variable m, are similar to step functions. An extrapolation algorithm that examined nearby nodes and extrapolated from the region of smallest variation could provide a better estimate of the state variables.

Another lingering question is the criteria for refinement of the adaptive grid. Previous investigations of adaptive grid algorithms have used the truncation error from the diffusion calculation to determine the new grid. It is necessary to repeat our experiment using this criterion and compare the results.

References:

1. Luo CH, Rudy Y. A dynamic model of the cardiac ventricular action potential.1. Simulations of ionic currents and concentration changes. *Circ Res.* 1994; 74: 1071-1096.

2. Ten Tusscher KHWJ, Noble D, Noble PJ, Panfilov AV. A model for human ventricular tissue. *Am J Physiol Heart Circ Physiol*. 2004; 286: H1573-H1589.

3. Jafri S, Rice JJ, Winslow RL. Cardiac Ca(2+) dynamics: the roles of ryanodine receptor adaptation and sarcoplasmic reticulum load. *Biophys J*. 1998; 74: 1149-1168.

4. Strang G. Numerical Analysis. *SIAM J.* 1969; 5: 526-517.

5. Kogan BY, Karplus WJ, Chudin EE. Heart fibrillation and parallel supercomputers. Paper presented at: Proceedings of International Conference on Information and Control, 1997; St. Petersburg, Russia.

6. Cherry EM, Greenside HS, Henriquez CS. A space-time adaptive method for simulating complex cardiac dynamics. *Phys Rev Lett*. 2000; 84(6): 1343-1346.

7. Cherry EM, Greenside HS, Henriquez CS. Efficient simulation of three-dimensional anisotropic cardiac tissue using an adaptive mesh refinement method. *Chaos*. 2003; 13(3): 853-865.

8. Chudin E, Goldhaber J, Garfinkel A, Weiss J, Kogan B. Intracellular Ca(2+) dynamics and the stability of ventricular tachycardia. *Biophys J.* 1999; 77(6): 2930-2941.

9. Chudin E, Garfinkel A, Weiss J, Karplus W, Kogan B. Wave propagation in cardiac tissue and effects of intracellular calcium dynamics (computer simulation study). *Prog Biophys Mol Biol.* 1998; 69(2-3): 225-236.

10. Winfree AT. Heart muscle as a reaction-diffusion medium: the roles of electrical potential diffusion, activation front curvature, and anisotropy. *Int J Bif & Chaos.* 1997; 7(3): 487-526.