Curvature Analysis of Cardiac Excitation Wavefronts

Abhishek Murthy, Ezio Bartocci, Flavio H. Fenton, James Glimm, Richard A. Gray, Elizabeth M. Cherry, Scott A. Smolka, and Radu Grosu

Abstract—We present the *Spiral Classification Algorithm* (SCA), a fast and accurate algorithm for classifying electrical spiral waves and their associated breakup in cardiac tissues. The classification performed by SCA is an essential component of the detection and analysis of various cardiac arrhythmic disorders, including ventricular tachycardia and fibrillation. Given a digitized frame of a propagating wave, SCA constructs a highly accurate representation of the front and the back of the wave, piecewise interpolates this representation with cubic splines, and subjects the result to an accurate curvature analysis. This analysis is more comprehensive than methods based on spiral-tip tracking, as it considers the entire wave front and back. To increase the smoothness of the resulting symbolic representation, the SCA uses weighted overlapping of adjacent segments which increases the smoothness at join points. SCA has been applied to a number of representative types of spiral waves, and, for each type, a distinct curvature evolution in time (signature) has been identified. Distinct signatures have also been identified for spiral breakup. These results represent a significant first step in automatically determining parameter ranges for which a computational cardiac-cell network accurately reproduces a particular kind of cardiac arrhythmia, such as ventricular fibrillation.

Index Terms—Cardiac excitation waves, isopotentials, Bézier curves, curvature, cardiac arrhythmia and fibrillation

1 Introduction

An estimated 81,000,000 American adults, more than one in three, have one or more types of cardiovascular disorders [19]. Among these, cardiac arrhythmias, such as ventricular tachycardia and especially fibrillation, may have devastating consequences (see Fig. 1 and [8]).

Determining the physiological conditions underlying a cardiac arrhythmia is a grand challenge, whose resolution may lead to innovative treatment strategies. An important component of this quest is the mathematical modeling, analysis, and simulation of cardiac-cell networks [7], [2], [14]. Among the myriad existing mathematical models of cardiac myocytes, Differential Equation Models of the reaction-diffusion type (DEMs) are arguably the most popular. In the context of DEMs, the above challenge can be reformulated as follows: *For what parameter ranges does a DEM network accurately reproduce the arrhythmia?*

The past two decades have witnessed the development of increasingly sophisticated DEMs [11], ranging from 4 to 87 state variables [5], [20], [23], [27], [15], [13]. The increase in the number of variables reflects the technological advances in capturing the intrinsic ionic mechanisms more accurately. The increase also leads to a simplification of the differential equations. For example, most of the equations of the 67-variable DEM of [15] are multiaffine, and were obtained using the law of mass action.

Unfortunately, the increase in the number of state variables inevitably leads to an increase in simulation time. In particular, simulation of the 67-variable DEM is so slow that its authors only simulated it in a single cell and even provided initial conditions corresponding to the steady state at different pacing rates to reduce the computation time for assessing the dynamics associated with those rates. In [2], we present CUDA-GPU implementations for many of the above-cited DEMs, on both Tesla and Fermi cards, with a dramatic reduction in simulation times. This allows us to perform, for the first time on a desktop computer, a 2D (surface) simulation of the 67-variable DEM.

The lowest dimensional DEM known to the authors that can accurately reproduce (experimentally) the macroscopic behavior of cardiac cells is the 4-variable Minimal model (MDM) of coauthors Fenton and Cherry [5]. The CUDA-GPU implementation of the MDM is so fast, that it allows the real-time simulation of a 500×500 grid of cells: one second of MDM simulation time is approximately equal to one second of real time.

- *A. Murthy and S.A. Smolka are with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-4400. E-mail: sas@cs.stonybrook.edu, amurthy@cs.sunysb.edu.*
- *E. Bartocci is with the Faculty of Informatics, Institute of Computer Engineering, Vienna University of Technology, Treitlstraβe 3, 1040 Vienna, Austria. E-mail: ezio.bartocci@tuwien.ac.at.*
- *F.H. Fenton is with the Department of Biomedical Sciences, Cornell University, T7012C Veterinary Research Tower, Ithaca, NY 14850. E-mail: flavio.h.fenton@cornell.edu.*
- *J. Glimm is with the Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794-3600. E-mail: glimm@ams.sunysb.edu.*
- *R.A. Gray is with the Food and Drug Administration, Office of Science and Engineering Laboratories, 10903 New Hampshire Avenue, WO62-Room 1114, Silver Spring, MD 20993-0002. E-mail: Richard.Gray@fda.hhs.gov.*
- *E.M. Cherry is with the Department of Biomedical Sciences, College of Veterinary Medicine, Cornell University, T7 012C Veterinary Research Tower, Ithaca, NY 14853. E-mail: elizabeth.m.cherry@cornell.edu.*
- *R. Grosu is with the Dependable Systems Group, Faculty of Informatics, Institute of Computer Engineering, Vienna University of Technology, Treitlestraβe 3, 1040 Vienna, Austria. E-mail: radu.grosu@tuwien.ac.at.*

Manuscript received 5 Dec. 2011; revised 29 Aug. 2012; accepted 31 Aug. 2012; published online 28 Sept. 2012.
Recommended for acceptance by F. Fages and S. Soliman.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBBSI-2011-12-0320.
Digital Object Identifier no. 10.1109/TCBB.2012.125.

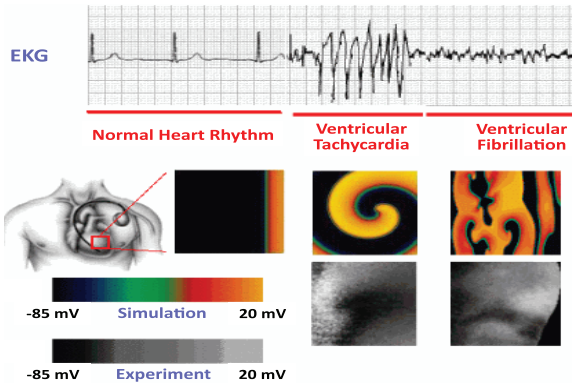


Fig. 1. Emergent behavior in cardiac-cell networks. Top: electrocardiogram. Middle and bottom: simulation and experimental mappings of spiral waves of electrical activity occurring in the heart during tachycardia and fibrillation.

This increase in the MDM simulation speed enables systematic exploration [25] of its associated parameter space. For example, in a recent student workshop on Computational Modeling and Analysis of Complex Systems, we asked the students in the workshop to repeatedly run the MDM model so that the ranges of two key MDM parameters, τ_w^- and τ_w^+, leading to fibrillation could be determined. Specifically, given a fixed spiral-initiation protocol, the students were asked to simulate the MDM model on each node of a 2D grid of parameter values, capture the generated waves, and track the tip movement of their spirals. Some of the results so obtained [3] are shown in Fig. 2.

This "Crowd Sourcing" approach provided very encouraging initial results. Crowd sourcing, however, is unlikely to scale up to the exploration of parameter spaces involving more than two parameters. Such exploration minimally requires: 1) A principled way of partitioning the parameter space; and 2) a fast and accurate algorithm for classifying spiral waves and their breakup.

In this paper, we present such a fast and accurate algorithm (SCA) for classifying spiral waves and their associated breakup. Given a digitized frame of a propagating wave, SCA constructs a highly accurate (with respect to the L2 error norm) representation of the front and the back of the wave, piecewise interpolates this representation with cubic splines, and subjects the result to an accurate curvature analysis (CA). This analysis is more comprehensive than spiral-tip tracking, as it considers the entire front and back of a wave. To increase the smoothness of the resulting symbolic representation, SCA uses a weighted overlapping of adjacent segments, which increases the smoothness at join points. To speed up the execution, we employ a GPU-CUDA implementation of a key component of the SCA algorithm, namely the isopotential reconstruction algorithm.

SCA has been applied to a number of representative types of spiral waves generated with the MDM (eight in total), and, for each type, a distinct curvature evolution in time (signature) has been identified. Distinct signatures have also been identified for the associated spiral breakup. Our SCA-based results, used in conjunction with parameter-estimation techniques, suggest a fully automatic approach to the grand challenge stated at the beginning of this section. Such an approach would be based on determining the parameter

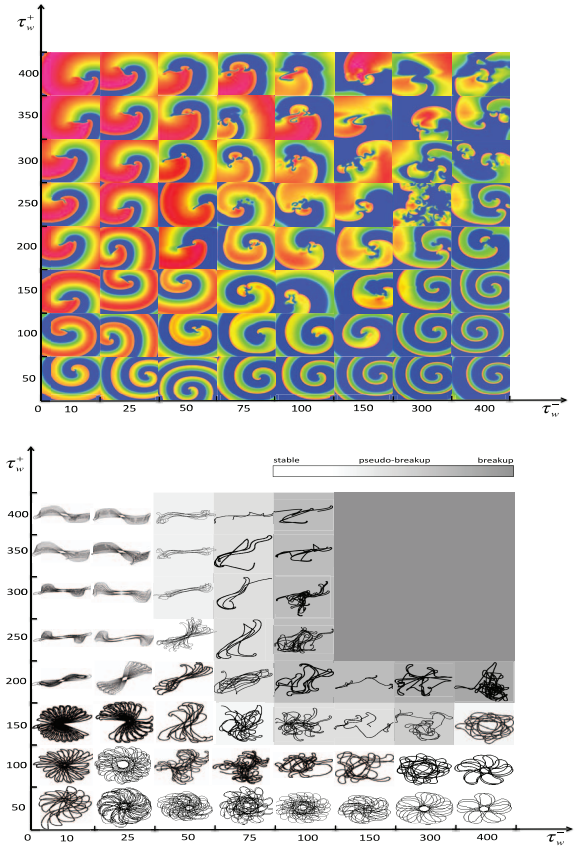


Fig. 2. Parameters τ_w^-/τ_w^+: (a) wave forms. (b) Tip movement, and regions of fibrillation.

ranges for which an MDM network accurately reproduces a particular form of cardiac arrhythmia. Assuming a connection has been established between model parameters and physiological entities, the parameter ranges so identified would reveal the root cause of the disorder and, concomitantly, a potential target for innovative treatment strategies.

The rest of the paper is organized as follows: Section 2 shows how we obtain the wavefront from a digitized frame. Section 3 considers isopotential curvature estimation. Section 4 presents our Bézier curve-fitting algorithm, while Section 5 considers a weighted-average-based improvement. Section 6 discusses the symbolic evaluation of the curvature along the isopotentials. Section 7 demonstrates the accuracy of the SCA algorithm on standard curves for which curvature can be calculated theoretically. Section 8 presents our case studies, while Section 9 surveys related work. Section 10 offers our concluding remarks and directions for future work.

2 Isopotential Reconstruction

The simulation data obtained by executing an $N \times N$ grid of DEMs, or the experimental data obtained by optically mapping a cardiac tissue with resolution $N \times N$, is a sequence of *digital frames* F_t of dimension $N \times N$. Each frame F_t is the snapshot at time t, with resolution $N \times N$, of the transmembrane *electrical potential* V_t, of the cardiac tissue.

The *wave fronts (wave backs)* of V_t are defined as the regions of V_t where each cell is in the activation (recovery) phase; that is, their electrical potential $V_t(x, y)$ equals

-30 mV, and their derivative $dV(x,y)/dt$ is positive (negative). The two fronts define together the *isopotentials* I_{-30} of V_t. These are curves of constant voltage in V_t, that never intersect each other. For simplicity, we will drop the subscript t occurring in F_t and V_t.

For any potential value v, the isopotentials I_v of V are smooth curves, whereas the isopotentials I_v of F are not. This is a consequence of the $N \times N$ resolution. The *isopotential reconstruction problem (IRP)* is therefore defined as follows: *Given a frame F, and a voltage value v, reconstruct the smooth I_v, isopotentials of V.* As we will see in the following sections, smoothness is essential for an accurate *curvature analysis* of the wave fronts and wave backs. The IRP remains the same when V is generalized to an arbitrary scalar field.

The *Isopotentials Reconstruction Algorithm (IRA)* is a key component of SCA, as it consumes most of the memory and time resources of the SCA. Hence, IRA has to be designed very carefully, and run on an appropriate hardware platform, to make the SCA a success. Today, the CUDA-GPUs of the NVIDIA' Fermi or Tesla video cards allow one to run a supercomputer on one's own desktop. We take advantage of these cards to implement a new *parallel, CUDA-based IRA (PIRA)*. To the best of our knowledge, this is the first algorithm of this kind.

PIRA belongs to a new CUDA-based class of algorithms that minimize the amount of synchronization; they require reconstruction of global information from a frame F. To achieve this goal, PIRA divides its work into: PMS, a fully parallel, local-information-computation procedure; PIE, a hierarchically parallel, global-information-computation procedure; and SOG, an optional, sequential to random-access, global-information-computation procedure.

2.1 Parallel Marching Squares (PMS)

The fully parallel, local-information-computation procedure PMS uses an adaptation of the *marching squares* algorithm. Given a frame F, PMS considers in parallel, that is in a different thread of a CUDA block, each 2×2 square s, of an adjacency-based $(N-1)\times(N-1)$ partition of F. Each thread locally and accurately computes the intersection points of zero, one or two lines with s. The number of lines and their crossing pattern with s depends on the values in the corners of s (type of s).

To determine the type of each square s of F, frame F is first subjected, in parallel, to the test $F_{i,j} \geq v$, for each index (i,j). The result of the test is stored in a Boolean matrix B. The Boolean value of the corners of s in B, traversed say in clockwise order, determine 16 possible intersection cases. These cases are shown in Fig. 3. In summary: 1) If all corners of s in B have the same value, there is no line crossing at all. 2) Otherwise, if diagonal corners have the same value, there are two ambiguous line crossings. Ambiguity is removed by averaging the value in all corners in s and comparing it with v. 3) Otherwise precisely one line is crossing s. Our adaptation also associates each with line, a direction by requiring that the zero corners of square s only occur to its left.

An isoline intersects s only between two corners that have opposite Boolean value in B. In other words, one corner has a value less than v and the other has a value which is greater than v. The intersection points, which also represent the starting and the ending points of a directed,

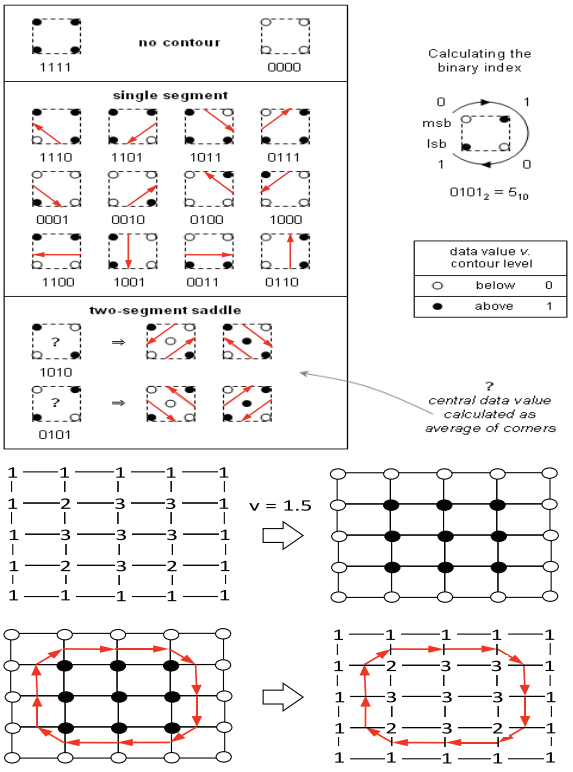


Fig. 3. Marching squares.

one-segment-long polyline, are computed via linear interpolation. For example, suppose that the corners of s at position (i,j) in B, result in bitvector 0111. Then, the line segment crossing s has the points $((x_0,y_0),(x_1,y_1))$ defined as follows:

$$x_0 = j, \quad y_0 = i + (v - F_{i,j}) \,/\, (F_{i+1,j} - F_{i,j})$$
$$y_1 = i, \quad x_1 = j + (v - F_{i,j}) \,/\, (F_{i,j+1} - F_{i,j}).$$

This information is stored in the corresponding one segment-polyline at position (i,j), in an $N \times N$ matrix S of squares. The starting and ending polylines of the open polylines list are initialized to the location of this polyline. If there are two polylines, the first one is the first in the list and the second one is the second. The list of closed polylines is initialized to null. The number of polyline segments and the number of open/closed polylines is also appropriately initialized.

```
struct Point        // typed points
{
   int    type;  // 0 = row crossing, 1 column crossing
   float  x,y;   // x and y coordinates
   float  y;     // y coordinate
}

struct PolyLine     // directed polylines
{
   int          type;        // 17 types
   int          number;      // number of segments
   Point        start, end;  // start and end points
   PolyLine *   next;        // ptr to next polyline
}
```

To speedup the interpolation process, the case analysis of the square type is efficiently prestored in a lookup table T, allocated in the constant memory of the GPU card.

```
struct Square                    // Data Structure for Squares
{
   int          num_of_opl;  // number of open polylines
   int          num_of_cpl;  // number of closed
                                polylines
   PolyLine *   opl_start;   // pointer to first open
                                polyline
   PolyLine *   opl_end;     // ptr to the last open
                                polyline
   PolyLine *   cpl_start;   // ptr to the first closed
                                polyline
   PolyLine *   cpl_end;     // ptr to the last closed
                                polyline
   PolyLine     poly_lines[2]; // storage allocated on leaf
                                level
}
```

For each square type t and for each one-segment polyline p, the table consists of the following entries: Two line coefficients and four coordinate displacements. For the above example, where $t = 7$, $p = 0$, table $T_{t,p}$ contains

$$\begin{aligned} T_{t,p} = \{&\{a = 0, b = 0, i_1 = 0, j_1 = 0, i_0 = 0, j_0 = 0\},\\ &\{a = 1, b = 0, i_1 = 1, j_1 = 0, i_0 = 0, j_0 = 0\},\\ &\{a = 1, b = 0, i_1 = 0, j_1 = 1, i_0 = 0, j_0 = 0\},\\ &\{a = 0\ b = 0, i_1 = 0, j_1 = 0, i_0 = 0, j_0 = 0\}\}. \end{aligned}$$

Denote $T_{t,0}$ and $T_{t,1}$ with the corresponding field names, indexed by subscripts 0 and 1. Then, the one-segment-polyline starting coordinate can be computed uniformly, as follows:

$$\begin{aligned} x_0 &= j + b_0 + a_0(v - F_{i+i_{00},j+j_{00}})/(F_{i+i_{01},j+j_{01}} - F_{i+i_{00},j+j_{00}})\\ y_0 &= i + b_1 + a_1(v - F_{i+i_{10},j+j_{10}})/(F_{i+i_{11},j+j_{11}} - F_{i+i_{10},j+j_{10}}). \end{aligned}$$

2.2 Parallel Isoline Extraction (PIE)

Using PMS one can readily plot the isolines of V. However, there is no global information available yet, despite the fact that we know the isoline segments, their orientation, and their linking. However, we do not know:

1. Which segments to the particular isolines?
2. How many isolines are there in V?
3. How many segments do they contain?
4. What are their starting and ending points.

The public *contour* function of Octave uses a sequential, recursive algorithm to obtain global information. This works as follows: 1) Traverse a matrix of interpolated segments, row by row and column by column, and pick the first unmarked one. 2) Then, recursively follow matching segments in a meaningful way, until the border or the starting point is reached again (no direction is readily available as in our PMS). During this process, dump all points traversed in a dynamic array, and mark all the segments as visited. 3) Once an entire isoline is found, search for the next unmarked segment, until the last row and column is reached.

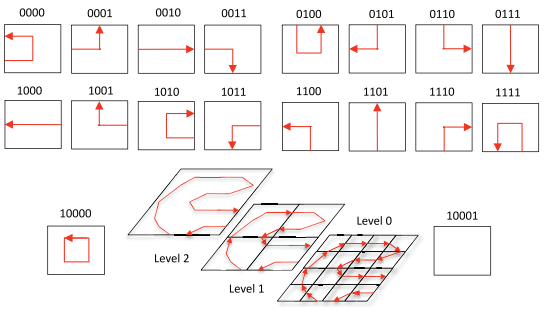


Fig. 4. Isopotential extraction.

Recursion and the sequential traversal of the entire matrix of segments is, however, prohibitive for a fast curvature analysis. The *isoline* function available at [21] is sequential, but not recursive. It uses the marching squares algorithm to compute all (disoriented) segments and dump them, in no particular order, in a dynamically allocated array. Then, it extracts the first unmarked segment, and repeatedly searches the entire array for an unmarked, matching continuation. Its time complexity is therefore comparable to the recursive algorithm.

To obtain a fast algorithm, we take advantage of the GPU cards. However, this imposes several important restrictions on PIE: 1) It cannot be recursive. 2) It cannot dynamically allocate memory. 3) It can only contain for-loops with known upper bounds. As shown in Fig. 4, the main idea of PIE is to organize the squares S passed by PMS in a quad-tree fashion, such that, at every level in the tree hierarchy, sibling nodes (and their immediate children) are processed in parallel, by a different CUDA thread. Hence, global information is computed sequentially in $log_4 N$ steps.

During this process, a parent either collects or pastes together the polylines of its children, if they have matching start and end points. A particular problem during this process is dynamic memory allocation. In the CUDA core, such allocation is not possible. Moreover, there is no way to predict in advance, without gross overestimation, the amount of memory needed in each node, to store the list of polylines it has identified so far, and the polylines themselves.

Our solution is to reuse the memory already allocated in the leaf squares for their two one-segment-long polylines during the upward sweep of the quad-tree. Each time, when information is collected from the four children, the result is stored in child 0. This information is later on passed upward in the hierarchy. Therefore, child 0's information gets lost.

Whenever the parent process matches the ending point of polyline p of child c with the starting point of polyline q of child d, it updates the starting and ending points of polyline pq in p, and removes q from the polyline list of child d. It then continues from a copy of q to find the next match. The total number of matches is bound by the number of polylines in all children, and this is used in a find-next-match for-loop. This loop is the equivalent of the recursive match-next-segment search in *contour* or *isoline*, but it is limited to the *polylines* of the parent's children. This dramatically decreases the time complexity of the recursive search.

Whenever the parent process collects the polyline lists of the children, it appends these lists to the lists of child 0. For

this purpose, the Square structure has the start/end fields of the list of open/closed polylines, and each polyline has a next polyline field. The number of polylines and the number of segments are also updated accordingly.

In order to efficiently match the end point of a polyline p of a child c with the starting point of a polyline q, it is important to know what sibling of c might have such a q. For this purpose, we classify the polylines according to their starting and ending faces in c, as shown in Fig. 4. This leads to a classification of polylines that is similar to the one for squares. In contrast to squares, however, polylines may start and end on the same face or they may be closed (that is, they are completely contained in c). The second case does not require further processing. Similarly, if a polyline of square c starts and ends on a face that is not adjacent to any of the c's siblings, no processing is required at this level either. To enable this kind of analysis, a type field is added to the polyline data structure, too.

The downside of reusing the squares while collecting and propagating information is that the linking information of one-segment-long polylines is lost. Consequently, one has to store this information in a different place. For simplicity and speedup reasons, we allocate two hash tables X and Y of size $N \times N$: the first is indexed by the integer value of horizontal intersection points, the second by the integer value of vertical intersection points. Each entry of X and Y stores the destination point and a bit classifying it as a horizontal or vertical intersection. This allows us to determine whether to choose the X or the Y hash table next. In general, unless the isolines are fractals, the number of segments is orders of magnitude smaller than $N \times N$. Hence, more information acquired about the kind of isolines to expect, may improve the size allocation, by choosing an $M << N$ for these tables.

2.3 Selection and Output Generation (SOG) Procedure

The optional SOG procedure selects the isolines of interest according to some given criterion, for example, the longest isoline, and stores the points of these isolines in an array, sorted by their traversal order. The size of the isolines, their starting point and their ending points, is available at the root of the quad-tree. This information is used to dynamically allocate the corresponding arrays outside of the CUDA core. The X and Y hash tables are then used to traverse the associated isoline and dump the points traversed in the array. This process transforms the local sequential-linking information, into a global, random-access information, where entry $i+1$ is known to be the successor point of entry i.

We have reconstructed the isolines of 10,000 frames, both with PIRA and the *contour* function of MATLAB. The first took 1.65 seconds while the second took 720 seconds. Hence, PIRA had a 444.44-fold speedup compared to *contour*. The platform was equipped with Intel Core i7-930 2.8 GHz LGA 1,366 130W Quad-Core Desktop Processor with OCZ Gold 12 GB (6×2 GB) 240-Pin DDR3 SDRAM DDR3 1600 (PC3 12800) Low Voltage Desktop Memory and a GPU Tesla C1060 with 240 SP cores divided equally among 30 SMs, and 4 GB of DRAM. The SP core clock speed was 1.29 GHz and the maximum bandwidth of memory access was 102 GB/sec. The rest of SCA has not been parallelized yet. However, this is easily accomplished, as we describe in the next sections.

3 Curvature Estimation

Propagation patterns of cardiac waves (isopotentials produced by SOG in the previous section) characterize cardiac arrhythmia. Obstacles or deformities affect the nominal linear motion of the waves resulting in anomalous propagation. Reentry is one such anomalous pattern in which cardiac waves assume spiral shapes, many of which are precursors to more dangerous arrhythmias. When these spiral waves break up, fibrillation, a possibly fatal arrhythmia follows. In [18], authors provide experimental evidence of atrial fibrillation being characterized by reentrant cardiac waves breaking up into chaotic patterns.

Local geometric features like curvature can be critical in affecting wave propagation and breakups. In [10], the authors establish the relation between the curvature of the isopotential and its propagation velocity. If θ_0 is the steady state velocity of a linear isopotential, then the velocity of a curved isopotential is given by $\theta = \theta_0 - \frac{D}{r}$, where r is the local radius of curvature and D is a coefficient determined by the passive properties of the cardiac medium. Thus, any part of the isopotential that has high positive curvature slows down and eventually breaks off. Also, the existing arrhythmia can be classified accurately based on the curvature of the underlying cardiac waves. Thus, an efficient method of measuring the evolving curvature would enable expressing emergent tissue-level behaviors. This would further help in predicting and identifying arrhythmias.

In reality, the cardiac tissue is strongly anisotropic, with waves propagating approximately three times faster along the direction of the heart muscle fibers (longitudinal) than in the transverse direction. This anisotropy can lead to problems in measuring true wave curvature, as noted in [28]. In practice, however, an anisotropic medium can be rescaled according to the known ratio of diffusion coefficients along and across the fibers to recover an isotropic medium. This rescaling can be used to extend our algorithm to anisotropic tissue behaviors. In [28, Section 4], other methods are discussed that can be used to adjust the measurements so that they can be compared with isotropic theory. We will therefore restrict our focus to isotropic media in the following discussion.

Any method that estimates the curvature of cardiac isopotentials must satisfy the following requirements: not only must it be highly accurate but the method must provide curvature values continuously along the perimeter of the isopotential. In other words, the method must be independent of the spatial resolution at which the isopotential was estimated or the grid on which the cardiac model was numerically integrated.

The workflow of our curvature estimation technique is shown in Fig. 5. The isopotential obtained in the previous section is a series of points on $\mathbb{R}^2$. Starting from this, our method of obtaining a smooth curvature estimate involves the following steps:

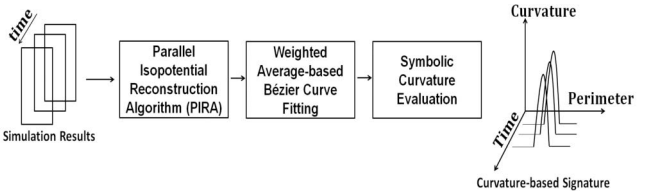


Fig. 5. Block diagram for curvature estimation.

1. **Step 0—Preprocessing.** Divides the isopotential obtained by SOG into overlapping strips of constant length.
2. **Step 1—Bézier curve fitting.** Fits each strip with a third degree Bézier curve, thus obtaining a piecewise degree-3 polynomial fit of overlapping Bézier curves.
3. **Step 2—Estimate curvature using symbolic analysis.** Curvature is calculated along the Bézier curves using symbolic calculations.
4. **Step 3—Obtain overall curvature estimate.** Obtains a smooth estimate of curvature along the entire isopotential.

First, we divide the isopotential I_v, obtained at potential v, into overlapping segments which we refer to as *strips*. The length of each strip is controlled by the parameter STRIP_LENGTH (SL). The percentage overlap among them is controlled by the parameter $OVERLAP$. This is the initial preprocessing done before the polynomial fitting. The following sections describe each of the remaining steps in detail.

4 Piecewise Bézier Fitting

Most of the formalisms of curvature in $\mathbb{R}^2$ involve second-order derivative terms. This motivates a degree-3 polynomial approximation to the isopotential. We fit the isopotential with overlapping cubic Bézier curves in a piecewise manner, thus satisfying the above-mentioned requirements of curvature estimation.

Let the jth strip of the isopotential I_v be denoted as $I_{v,j}$. We obtain one Bézier curve that approximates this strip up to a certain degree of L2 error. A Bézier curve approximation for $I_{v,j}$ would have the following parametric form, where $t \in [0,1]$:

$$X_j(t) = (1-t)^3 P_j^0 + 3t(1-t)^2 P_j^1 + 3t^2(1-t)P_j^2 + t^3 P_j^3 \quad (1)$$

$$Y_j(t) = (1-t)^3 Q_j^0 + 3t(1-t)^2 Q_j^1 + 3t^2(1-t)Q_j^2 + t^3 Q_j^3. \quad (2)$$

Here, $P_j^0 - P_j^3$ and $Q_j^0 - Q_j^3$ are the control points of the Bézier curves, which approximate the isopotential strip along x- and y-axes, respectively. To sample from this curve, we evaluate the above-mentioned expressions over the interval $t \in [0,1]$. *Terminal* control points P_j^0, P_j^3, Q_j^0, and Q_j^3 coincide with the start and end points of the strip, respectively. The fitting procedure, adapted from [17] and [24], optimizes the intermediate control points P_j^1, P_j^2, Q_j^1, and Q_j^2 to minimize the least squared error. We assume uniform parametrization of t in $[0,1]$ for each segment. The error functions for fitting $I_{v,j}$ are

$$E_x = \sum_{i=1}^{SL} [x_i - X_j(t_i)]^2, E_y = \sum_{i=1}^{SL} [y_i - Y_j(t_i)]^2,$$

where (x_i, y_i) denotes the ith point on the isopotential strip. On replacing (1) and (2) in the above error functions, respectively, we obtain

$$\sum_{i=1}^{SL} \left[x_i - (1-t_i)^3 P_j^0 - 3t(1-t_i)^2 P_j^1 - 3t_i^2(1-t_i)P_j^2 - t_i^3 P_j^3\right]^2,$$
$$\sum_{i=1}^{SL} \left[y_i - (1-t_i)^3 Q_j^0 - 3t_i(1-t_i)^2 Q_j^1 - 3t_i^2(1-t_i)Q_j^2 - t_i^3 Q_j^3\right]^2,$$

as expressions for E_x and E_y, respectively. We show the calculations only for E_x. For E_y, the expressions follow from those for E_x. Control points P_j^1 and P_j^2 can be obtained at the minimum value of E_x by

$$\frac{\partial E_x}{\partial P_j^1} = 0, \frac{\partial E_x}{\partial P_j^2} = 0.$$

Solving the above two equations we obtain the following expressions for P_j^1 and P_j^2:

$$P_j^1 = \frac{\alpha_2^j \beta_1^j - \alpha_3^j \beta_2^j}{\alpha_1^j \alpha_2^j - \alpha_3^{j\,2}}, P_j^2 = \frac{\alpha_1^j \beta_2^j - \alpha_3^j \beta_1^j}{\alpha_1^j \alpha_2^j - \alpha_3^{2j}},$$

where, $\alpha_1, \alpha_2, \alpha_3, \beta_1$ and β_2 for the jth segment are given by

$$\alpha_1 = 9\sum_{i=1}^{SL} \left[t_i^2(1-t_i)^4\right],$$
$$\alpha_2 = 9\sum_{i=1}^{SL} \left[t_i^4(1-t_i)^2\right],$$
$$\alpha_3 = 9\sum_{i=1}^{SL} \left[t_i^3(1-t_i)^3\right],$$
$$\beta_1 = 3\sum_{i=1}^{SL} \left[t_i(x_i - (1-t_i)^3 P_0 - t_i^3 P_3)(1-t_i)^2\right],$$
$$\beta_2 = 3\sum_{i=1}^{SL} \left[t_i^2(x_i - (1-t_i)^3 P_0 - t_i^3 P_3)(1-t_i)\right].$$

The fitting procedure described above is sequential in nature. However, for all the strips, one can run this procedure in parallel, by using a different CUDA-thread on the GPU cards. Although we have not done this yet, this is a simple adaptation of this algorithm. To make the curvature estimates smooth, we fit the curves on overlapping segments, as explained in the next section.

5 Weighted Average-Based Improvement

After each strip is fit with cubic Bézier curves as explained above, we improve the smoothness of the overall fit. A pure piecewise fitting approach would create discontinuities in the derivative of the isopotential approximation at the points where two curves meet. We ensure that derivatives up to second order are well defined everywhere on the isoline. To do this, fitting is performed on overlapping strips. Parameter OVERLAP determines the percentage overlap between adjacent strips of the isopotential.

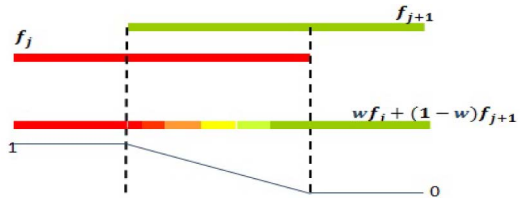


Fig. 6. Weighted average-based fit smoothing and curvature estimation.

Consider a part of the isopotential that has two adjacent overlapping strips with indices j and $j+1$. To define the Bézier curve fit for the part of the isoline covered by the two strips, we use a weighted average-based method. Suppose the curves describing the two strips are f_j and f_{j+1}. Then, the fit for the region covered by the two strips is given by $w_j f_j(t_j) + w_{j+1} f_{j+1}(t_{j+1})$, where $w_{j+1} + w_j = 1$. In essence, the influence of the adjacent curves on the fit is gradually varied in the region of overlap. In the region where there is no overlap, the fit is completely described by the only Bézier curve corresponding to that strip. As one enters the region of overlap, the fit is a weighted average of the two Bézier curves corresponding to adjacent strips. The weights are varied linearly in our scheme, as shown in Fig. 6. The same weighted-average smoothing is performed for the derivatives and the curvature values.

Each Bézier curve is a third degree polynomial. In the overlapping region, the fit is a weighted average of two cubic polynomial functions. Thus, even in the overlapping region the fit is C^2 smooth, rendering it amenable to curvature estimation (described in the next section). The smoothing described above can also be performed in parallel. The weighted-average based scheme is different from [16], which uses cubic splines to fit the isopotentials.

6 Curvature Estimation Using MATLAB Symbolic Computation

We use symbolic computations in MATLAB to evaluate curvature along the isopotential. This constitutes step 2 of our method listed above. The fitting procedure described above provides a set of smooth Bézier curves that describe the isopotentials. As these are closed form expressions, they can be manipulated symbolically to calculate the magnitude of the curvature along the isopotentials. MATLAB's symbolic toolbox provides the facility to declare symbolic variables, construct functions out of them and operate on those functions. Once the operations yield the expressions of interest, they can be evaluated at arbitrary resolution by suitably specifying the interval for the symbolic variables.

In our case, we obtain the functions $X_j(t)$ and $Y_j(t)$ for each strip. The absolute curvature of the strip of the isopotential is derived using elementary calculus

$$\kappa_j(t) = \frac{|r'_j(t) \times r''_j(t)|}{|r'_j(t)|^3}, \tag{3}$$

where, $r_j(t) = [X_j(t), Y_j(t)]$ is the position vector described by the Bézier curve. The grid size for all the cardiac simulations is in the order of microns, thus, the unit of the curvature measured is $(\mu m)^{-1}$. The important point to note is that $X_j(t)$, $Y_j(t)$ and thus $r_j(t)$ are managed as symbolic expressions which are functions of the symbolic variable t. Thus, $\kappa_j(t)$ is obtained in closed form as an expression in t. Symbolic operations on $r_j(t)$ are performed using MATLAB's symbolic math toolbox [22].

After obtaining closed form expressions for curvature, their continuity ensures that we can evaluate them at any resolution of the parameter t. This translates to obtaining a continuous estimate of curvature along the perimeter of the isopotential.

Step 3 of our method evaluates these curvature functions along the isopotential using the weighted approach described in the previous section. Currently, we maintain uniform resolution for t along all the strips. Adapting this to the shape of the isoline is part of our future work. In particular, the information stored in the quad-tree of PIE, for example, the filling factor of the area associated with its nodes might facilitate a fast and accurate breakup of the isoline in isoline strips, improving on the idea in [16].

An alternative approach to curvature estimation involves precomputing the generic form of the curvature function for a Bézier curve. Consider a curve $r(t) = [X(t), Y(t)]$ where $X(t)$ and $Y(t)$ are described by Bézier curves in (1) and (2), respectively. Also let $a_x = -P_j^0 + 3P_j^1 - 3P_j^2 + P_j^3, b_x = 3P_j^0 - 6P_j^1 + 3P_j^2, c_x = -3P_j^0 + 3P_j^1$, $d_x = P_j^0$ and a_y, b_y, c_y, d_y be defined symmetrically. Expressions $a_x \ldots d_x$ and $a_y \ldots d_y$ define the curves $X(t)$ and $Y(t)$ in the nominal polynomial form, for example, $X(t) = a_x t^3 + b_x t^2 + c_x t + d_x$. Now

$$\begin{aligned} |r'(t) \times r''(t)| &= X'(t).Y''(t) - Y'(t).X''(t) \\ &= [12(b_x a_y - a_x b_y) + 6(a_x b_y - b_x a_y)]t^2 \\ &\quad + [6(c_x a_y - a_x c_y)]t + 2[b_y c_x - b_x c_y]. \end{aligned}$$

In the same way, we can calculate

$$\begin{aligned} |r'(t)| &= [r'(t).r'(t)]^{1/2} \\ &= [(3a_x t^2 + 2b_x t + c_x)^2 + (3a_y t^2 + 2b_y t + c_y)]^{1/2}. \end{aligned}$$

Using the above expressions, the curvature of each strip can be calculated. This method is also amenable to parallel computation as it does not need the symbolic computational toolbox of MATLAB. We have implemented both techniques. The curvature for the whole isoline is calculated using the weighted average-based method as explained above.

The runtimes for the curve fitting and curvature calculation routines depend on the length of the isopotential. The extraction process checks each grid square for a possible point of the isopotential. The number of edges on a $n \times n$ grid can be calculated by solving the following recurrence:

$$\begin{aligned} E(n) &= E(n-2) + 4n + 4(n-3) + 8 \\ E(1) &= 4, E(2) = 8. \end{aligned}$$

The solution is given by

$$E(n) = -2(-1)^n + 2n(n+1) - 2.$$

Hence the maximum length of the isoline is $O(n^2)$, which bounds the number of operations in the fitting and curvature routines to $O(n^2)$.

TABLE 1
Curvature Estimation for Standard Curves, $a = 15$, Total Number of Points on Each Curve = 4,000, $SL = 150$, $OVERLAP = 70$

Curve in polar co-ordinates (r, t)	Curvature $\kappa(t)$	Average L2 error: fit	Average percentage L2 error: κ
Archimedes' spiral: $r = at$	$\frac{2+t^2}{a(1+t^2)^{3/2}}$	0.0515	1.62
Hyperbolic spiral: $r = a/t$	$\frac{t^4}{a(1+t^2)^{3/2}}$	0.0011	1.88
Circle: $r = a$	$1/a$	0.0073	0.76
Parabola $r = \frac{-2}{1+cost}$	$\frac{1}{2(1+t^2)^{3/2}}$	0.0025	8.1

7 Curvature Estimation Accuracy

The curvature estimation technique developed in Sections 3, 4, 5, and 6 was implemented on data generated by plotting standard curves, for which curvature can be calculated in closed form. A summary is given in Table 1.

The standard curves were generated by plotting them in MATLAB. Then, their fits were obtained using the weighted average-based Bézier curve fitting, and curvature was computed along the perimeter of the curve. The error depends on the granularity of the data; if more points are sampled from the theoretical curve and then subjected to curvature estimation, the accuracy increases. Figs. 7a, 7b, 7c, 7d, 7e, 7f, 7g, and 7h show the results of these experiments.

8 Case Studies

To demonstrate our isopotential tracking and curvature estimation techniques, we simulated six different types of tachycardia and one case of fibrillation. In an additional case study, the spiral waves resulting due to an obstacle in the tissue were investigated. We applied isopotential extraction on regularly spaced (in time) frames from the simulation results. Then, we estimated the curvature along the cardiac wave to capture the characteristic trend in time. In the following section, we show for each case study:

1. one typical frame from the simulation results,
2. the corresponding isopotential that was tracked,
3. weighted average-based Bézier curve fit, and
4. curvature trend, in time, for that simulation.

We refer to the curve traced by the wave, along which the curvature was measured, as the wave's *perimeter*. The curvature trend plots the curvature of the wave along its perimeter, as it evolved in time. The perimeter for our case studies is measured in microns, as the simulations were performed at that scale. Curvature is therefore measured in $(\text{microns})^{-1}$ units. We used the Barkley model [1] in the first case study and thus the spatial dimension (and therefore the perimeter and the curvature) is unitless. It should be noted that to generate case study 4) we plot the closed-form function corresponding to curvature. The function itself can be evaluated at any resolution of the Bézier curve parameter t.

Our case studies support two findings. First, we demonstrate the correctness of the isopotential extraction and the curvature estimation techniques explained above. Second, the results show the feasibility of using the curvature trend as a feature to classify different arrhythmic spiral excitation waves. For this section, we define the tip of a spiral-shaped cardiac wave as the point with the highest curvature and it separates the wavefront and the *waveback* regions of the wave. The case studies were generated such that the tips of the spirals trace trajectories of different shapes resulting in different types of abnormal propagation.

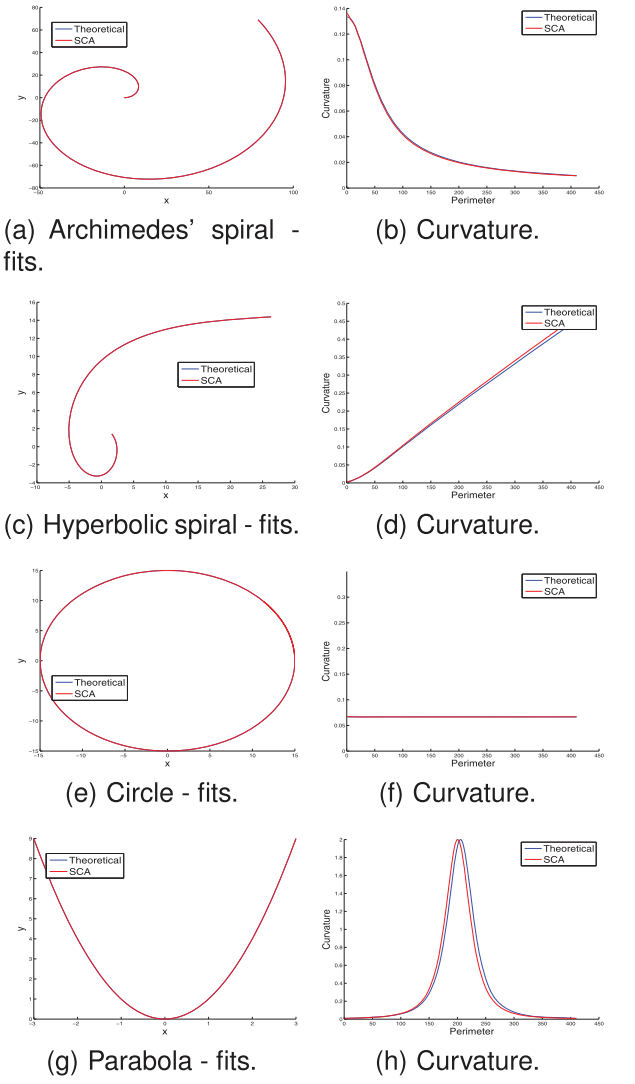
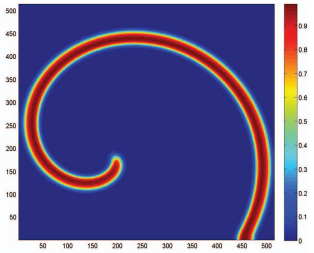


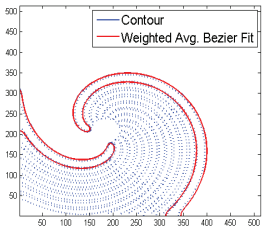
Fig. 7. Accuracy experiments

Case 1. Reentry with circular core: In this case study, we simulated the Barkley model [1] on a tissue of 514×514 cells. Simulation frames were processed at a rate of once every 10 ms in simulation time, i.e., the frames were sampled at the rate of once every 10 ms, from the simulation. The core of the spiral shaped excitation waves traced a circular trajectory. One sample frame of simulation result is shown in Fig. 8a. After the initial excitation, the spiral-shaped isopotentials were extracted at a scaled level of $u = 0.7$ (where u is a state variable of the Barkley model).

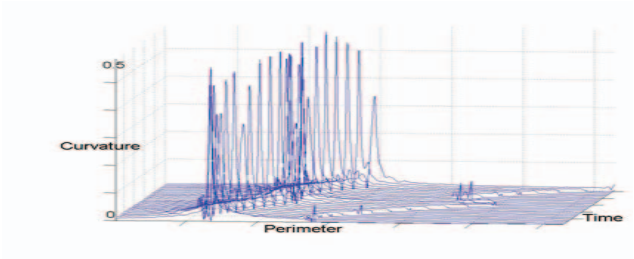
Fig. 8b shows the propagation of the isopotential with the tip tracking a circular trajectory. The waves in the first and last time steps of simulation are shown in solid blue. The overlapping red curve is the weighted average-based



(a) Simulation, colorbar: unitless scaled potential u of the Barkley model [1].



(b) Isopotential evolution.



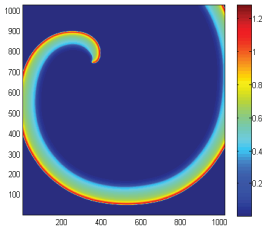
(c) Curvature Trend.

Fig. 8. Results for case study 1: reentry with circular core. For the curvature trend, the perimeter and the curvature are unitless and time is in steps of 10 ms.

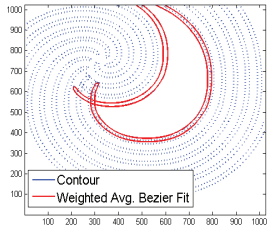
Bézier fit. Isopotentials are shown for intermediate frames in dotted blue. As the spiral rotates, its tip tracing a circular trajectory, the basic shape of the spiral isopotential does not change. As the basic shape of the wave remains unchanged, we would expect the curvature trend to remain the same over time. The curvature trend is shown is Fig. 8c. The region of maximum curvature corresponds to the spiral tip which remains around the center of the isopotential throughout the simulation. The rest of the isopotential shows a relatively lower curvature. This trend characterizes circular-core reentrant spiral waves in the heart.

Case 2. Reentry with hypocycloidal core: It is not always the case that the tip is located around the center of the spiral wave. To simulate asymmetric shapes, we generated waves whose tips trace hypocycloidal trajectories. In this case study, a tissue of size $1{,}024 \times 1{,}024$ was simulated using the Minimal model [5]. The isopotential was extracted again for $u = 0.7$ (where u is a state variable of the Minimal model) and the rate at which the frames were processed was once every 10 ms. A typical frame of the simulation can be seen in Fig. 9a. Both the isopotentials extracted and the Bézier fits are shown for the first and the last frames of the simulation. The intermediate dotted blue lines are isopotentials for some of the intermediate frames.

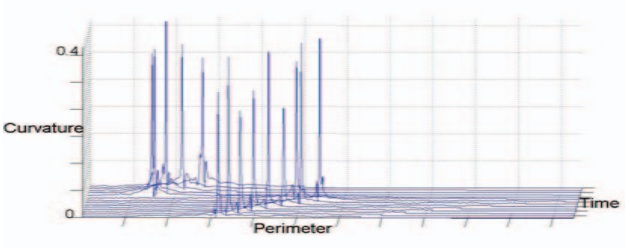
As the tip rotates, the length of the wave changes and at times, the tip is not the center of the isopotential. This turning causes the shape of the wave to become asymmetric. The turn of the spiral is evident in the curvature trend in Fig. 9c. As the length of the spiral changes, the region of highest curvature shifts on the curvature trend. This further demonstrates that a trend of morphological features like curvature, can capture the dynamics of different types of cardiac arrhythmias.



(a) Simulation, colorbar: unitless scaled potential u of the Minimal model [5].



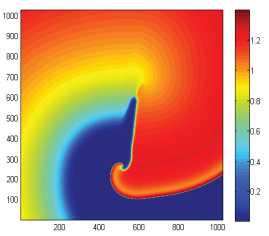
(b) Isopotential evolution.



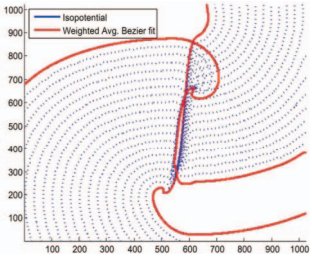
(c) Curvature Trend.

Fig. 9. Results for case study 2: reentry with hypocycloidal core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu\text{m})^{-1}$ and time is steps of 10 ms.

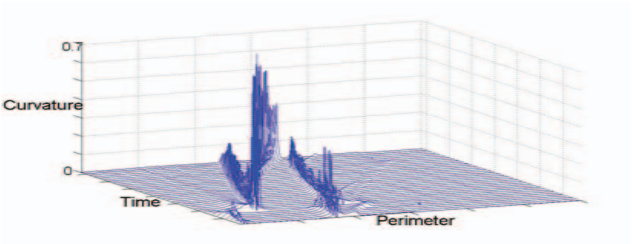
Case 3. Reentry with linear core: In practice, spiral waves may exhibit more complex behavior. In the presence of an obstacle or deformity in the medium, the rotating waves may assume linear trajectories. We studied such waves in this case study by simulating a tissue of $1{,}024 \times 1{,}024$ cells using the Minimal model [5] and processed the simulation frames at a rate of once every 10 ms. Isopotential extraction was done for a scaled level of $u = 1.0$. Fig. 10a shows a



(a) Simulation, colorbar: unitless scaled potential u of the Minimal model [5].

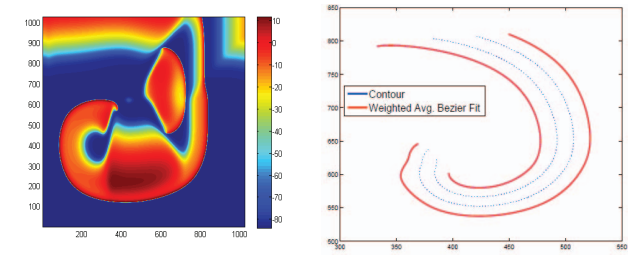


(b) Isopotential evolution.

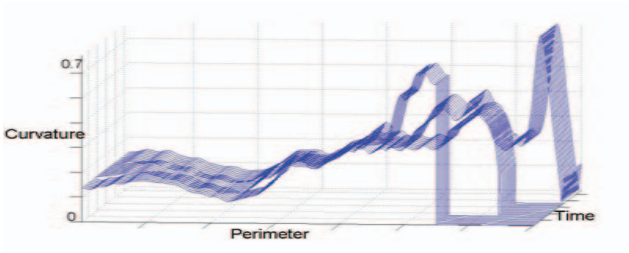


(c) Curvature trend.

Fig. 10. Results for case study 3: Reentry with linear core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu\text{m})^{-1}$ and time is in steps of 10 ms.

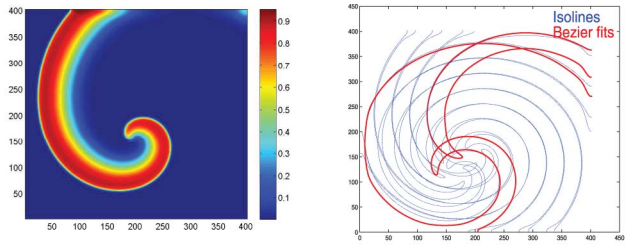


(a) Simulation, colorbar - membrane potential (mV). (b) Isopotential evolution.

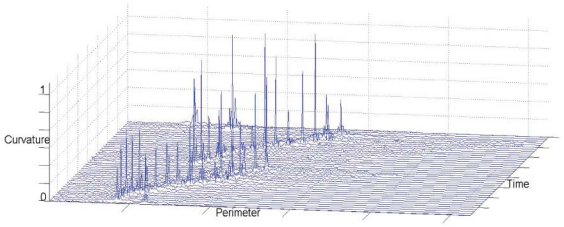


(c) Curvature trend.

Fig. 11. Results for case study 4: spiral break up. For the curvature trend, perimeter is measured in μm, curvature in $(\mu\mathrm{m})^{-1}$ and time is in steps of 1 ms.



(a) Simulation, colorbar: unitless scaled potential u of the 3-variable model [12]. (b) Isopotential evolution.



(c) Curvature trend.

Fig. 12. Results for case study 5: cycloidal core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu\mathrm{m})^{-1}$ and time in steps of 10 ms.

snapshot of the simulation. The linear trajectory along which the wave rotates can be seen with the tip of the wave at one end. Fig. 10b shows the evolving isopotential. It starts from the solid isopotential line of Fig. 10b and ends at the other solid isoline.

As shown in Fig. 10c, during its linear motion, the wave has three regions of high curvature. The first two are present at the ends of the linear path of the tip. As always, the highest curvature is found at the tip which separates the wavefront and the wave back. As the tip moves along the linear path, the separation between curvature peaks corresponding to the high curvature regions, changes. The highest peak starts near the left peak that corresponds to the first high curvature bend of the isopotential. With time, as the tip moves down to the other end of the linear path, the central curvature peak shifts toward the right peak.

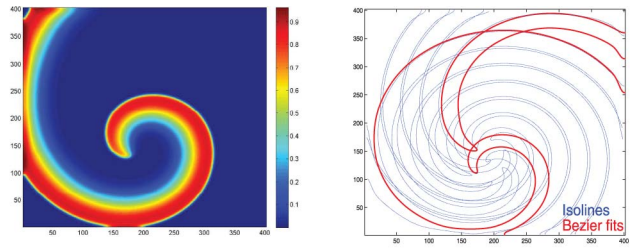
Case 4. Spiral wave breakup: We study the onset of fibrillation in this case study. The spatiotemporal definition of the fibrillating myocardium involves the breakup of reentrant waves [18]. This breakup creates spirals which interact to produce emergent behavior. Thus, predicting the occurrence of spiral breakup is crucial to the problem of predicting fibrillation. A tissue of $1{,}024 \times 1{,}024$ cells was simulated using the Beeler-Reuter model [4]. Spiral breakup occurs at a very short time scale. Therefore, the frames (output) of the simulation were processed at the rate of once every 1 ms.

Fig. 11a shows one simulation frame, where the first breakup has already occurred. We tracked the isopotential of value $V = -3\ \mathrm{mV}$ (where, V is the membrane potential variable in the model) until the first breakup occurred. As we approached the moment of detachment, the isopotential showed a dent near the site of break up. This change in shape translated to the creation of a high curvature region. The changing shape of the isopotential is shown in Fig. 11b. Again, the isopotential and the polynomial fits are shown for the first and last step, and the intermediate steps are shown in dashed lines. As compared to other studies, there seem to be fewer intermediate curves. This is because, with a small time step of 1 ms, most of the frames do not show any change, leading to overlapping isopotentials. The time scale at which detachment occurs forced us to increase the processing rate of the frames of the simulation.

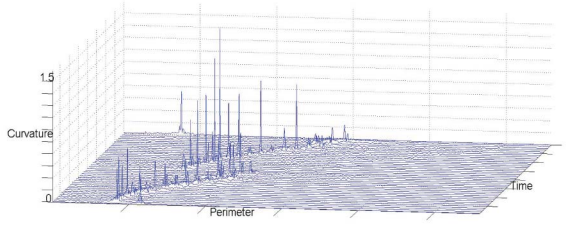
Fig. 11c shows the trend of curvature as the isopotential evolves toward breakup. Just before detachment, we see high curvature corresponding to the evolving site of breakup. Thus, the gradual build up of a high curvature site on the wave is a strong indication of future breakup leading to eventual fibrillation.

Case 5. Cycloidal core: A tissue of 402×402 cells was simulated using the 3-variable model of [12] and the frames were processed at the rate of once every 10 ms. The tip of the spiral followed a cycloidal trajectory. In this process, the length of the contour, estimated at a normalized level of $u = 0.7$ (where u is a state variable of the model), did not remain constant. The trajectory traversed by the tip can be observed in the isoline evolution shown in Fig. 12b. The weighted average-based fit is shown only for the first and last frames. As shown in Fig. 12c, a region of high curvature is found near the tip of the spiral. The changing position of the curvature peak reflects the path traversed by the tip along its trajectory.

Case 6. Epicycloidal core: In this case study, a tissue of 402×402 cells was simulated using the 3-variable model of [12], using a processing rate of once every 10 ms. In this case, the tip follows an epicycloidal trajectory. Fig. 13a shows a typical frame of the simulation. The isopotential and curvature evolution show the traversal of the tip of the

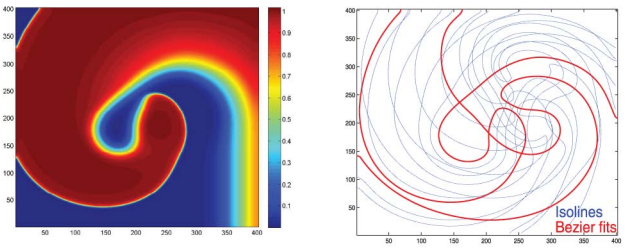


(a) Simulation, colorbar: unitless scaled potential u of the 3-variable model [12]. (b) Isopotential evolution.

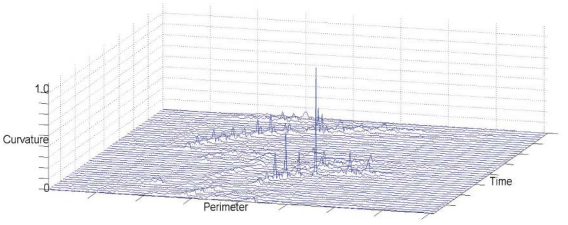


(c) Curvature trend.

Fig. 13. Results for case study 6: epicycloidal core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu\mathrm{m})^{-1}$ and time is in steps of 10 ms.



(a) Simulation, colorbar: unitless scaled potential u of the 3-variable model [12]. (b) Isopotential evolution.



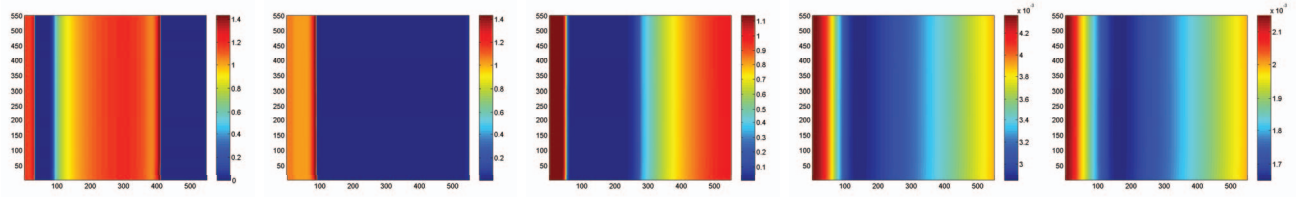
(c) Curvature trend.

Fig. 14. Results for case study 7: hypermeandering core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu\mathrm{m})^{-1}$ and time is in steps of 10 ms.

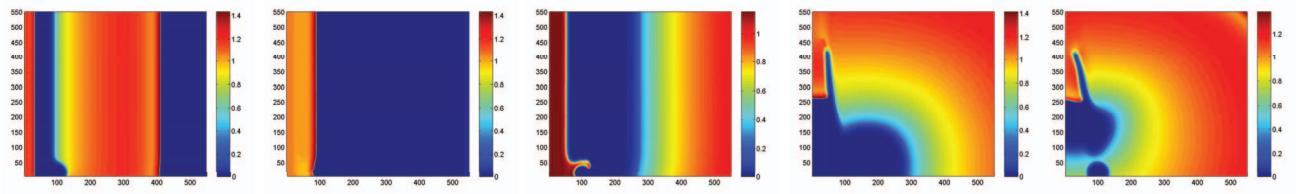
spiral along an epicycloidal path. Isolines were calculated for a scaled level of $u = 0.7$ for this case study.

Case 7. Hypermeandering core: Hypermeandering spirals involve irregular motion of the tip along various paths. For this case study, a tissue of 402×402 cells was simulated using the 3-variable model of [12] and the frames were processed at the rate of once every 10 ms. The isolines were estimated at a scaled level of $u = 0.9$. At this level, the spiral shape of the contour is not evident initially. This is reflected in the flat curvatures shown in Fig. 14c. The starting isoline and its fit are diagrammed in Fig. 13b. As this isoline moves along an unpredictable path, a spiral shape appears and results in high curvature peaks in later curvature plots in Fig. 14c.

Case 8. Obstacle-induced reentry: This case study explored the curvature trend of reentrant waves that develop due to tissue inhomogeneities. We simulated a tissue of $1{,}038 \times 1{,}038$ cells using the Minimal model [5] in two different settings. First, all the cells were simulated under nominal conditions, which resulted in planar excitation wavefronts shown in Figs. 15a, 15b, 15c, 15d, and 15e. Then, the cells lying in a circular region, were made inactive by decoupling them from the tissue, resulting in inhomogeneity. The inactive cells do not participate in the propagation of waves, and cause them to slow down. This results in spiral waves shown in Figs. 15f, 15g, 15h, 15i, and 15j. We only plot a part of the tissue consisting of 550×550 cells and some time steps (t) of the simulation.



(a) t = 17. (b) t = 49. (c) t = 81. (d) t = 113. (e) t = 145.



(f) t = 17. (g) t = 49. (h) t = 81. (i) t = 113. (j) t = 145.

Fig. 15. Case study 8: simulation frames for planar ((a) through (e)) and obstacle-induced reentrant waves ((f) through (j)).

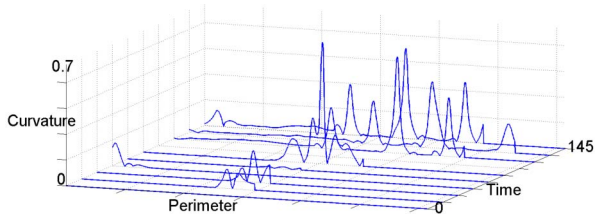


Fig. 16. Curvature trend of reentry setting in due to tissue inhomogeneity, case study 8. The perimeter is measured in μm, curvature in $(\mu\mathrm{m})^{-1}$ and time in steps of 10 ms.

The curvature of the waves was recorded using our algorithm, as they transformed from planar to spirals. Fig. 16 plots the curvature trend for isopotentials of value $u = 0.7$. Initially, when the waves are planar the curvature is 0. The obstacle causes the slowing down of a part of the wave, resulting in small curvature peaks. These peaks developed into sustained patterns as the obstacle-induced arrhythmia sets in. The frames of the simulation were processed at the rate of once every 10 ms.

9 Related Work

Previous work of one of the authors, [16], is most closely related to the proposed algorithm. There are two main differences. First, our GPU-based approach of identifying and extracting isolines in each frame is new. This enables fast processing of simulation results for curvature analysis. The second difference lies in the piecewise-smooth polynomial approximation performed to estimate the curvature. The authors in [16] use splines to approximate the isoline. The uniform-length strips and their cubic Bézier curve fits used in SCA are amenable to GPU-based parallel methods.

The wave curvature and the refractory period of the cardiac cells influence the motion of cardiac waves. The role of curvature in wave propagation was reported in [10]. The relation between wave curvature and velocity explained in Section 3, is analyzed in this work, specifically for the meandering of spiral waves. The authors in [6] show that the curvature of a propagating wave front can cause slow conduction and block in normal and homogeneous tissue. They identify conditions based on curvature that would lead to wave detachment.

In [9], the effect of curvature on propagation speed, the action potential duration, and the refractory period is studied. In the recent work of [26], multiple spirals and their interactions were investigated. The authors report that the velocity-curvature relation might not apply directly to regions with altered source-sink relationships, which involves sites of wave collisions and spiral cores.

10 Conclusions

Technological developments within the graphics processing community, NVIDIA in particular, coupled with theoretical advances in the computer-aided verification community, have set the stage for fast simulation, powerful analysis, and accurate prediction of complex biological processes. In this paper, we have presented a key component of such a framework: a parallel curvature analysis algorithm that given a series of frames generated via simulation or optical mapping produces a curvature-based signature of the wave/spiral captured in the frames. Our isopotential reconstruction algorithm takes advantage of NVIDIA's Tesla and Fermi graphics processing cards, and the associated CUDA architecture. Our results demonstrate speed-up by a factor of 444.44 for isopotential reconstruction compared to the MATLAB-based contour algorithm. Our case studies identified distinct signatures for various forms of cardiac arrhythmias (eight in total), which may be used to classify a wave by its spiral type. We plan to implement accurate classifiers that would use the curvature-based signature as the main discriminating feature to label a given simulation with one of the arrhythmias encountered in the training phase. We are currently working on parallelizing the curve fitting and signature-generation components of the SCA algorithm and further expanding the set of case studies.

Acknowledgments

This research was performed under the US National Science Foundation (NSF) Expeditions project on Computational Modeling and Analysis for Complex Systems (CMACS) funded by the grant: NSF CCF-0926190 (http://cmacs.cs.cmu.edu).

References

[1] D. Barkley, "A Model for Fast Computer Simulation of Waves in Excitable Media," *Physica*, vol. 49, nos. 1/2, pp. 61-70, Apr. 1991.

[2] E. Bartocci, E.M. Cherry, J. Glimm, R. Grosu, S.A. Smolka, and F.H. Fenton, "Toward Real-Time Simulation of Cardiac Dynamics," *Proc. Ninth Int'l Conf. Computational Methods in Systems Biology (CMSB '11)*, pp. 103-112, 2011.

[3] E. Bartocci, R. Singh, F.B. von Stein, A. Amedome, A.J.J. Caceres, J. Castillo, E. Closser, G. Deards, A. Goltsev, R.S. Ines, C. Isbilir, J.K. Marc, D. Moore, D. Pardi, S. Sadhu, S. Sanchez, P. Sharma, A. Singh, J. Rogers, A. Wolinetz, T. Grosso-Applewhite, K. Zhao, A.B. Filipski, R.F. Gilmour, R. Grosu, J. Glimm, S.A. Smolka, E.M. Cherry, E.M. Clarke, N. Griffeth, and F.H. Fenton, "Teaching Cardiac Electrophysiology Modeling to Undergraduate Students: Laboratory Exercises and GPU Programming for the Study of Arrhythmias and Spiral Wave Dynamics," *Advances of Physiology Education*, vol. 35, no. 4, pp. 427-437, Dec. 2011.

[4] G. Beeler and H. Reuter, "Reconstruction of the Action Potential of Ventricular Myocardial Fibres," *J. Physiology*, vol. 268, no. 1, pp. 177-210, June 1977.

[5] A. Bueno-Orovio, M.E. Cherry, and F.H. Fenton, "Minimal Model for Human Ventricular Action Potentials in Tissue," *J. Theoretical Biology*, vol. 253, no. 3, pp. 544-560, 2008.

[6] C. Cabo, A.M. Pertsov, J.M. Davidenko, and J. Jalife, "Electrical Turbulence as a Result of the Critical Curvature for Propagation in Cardiac Tissue," *Chaos*, vol. 8, no. 1, pp. 116-126, Mar. 1998.

[7] E.M. Cherry and F.H. Fenton, "A Tale of Two Dogs: Analyzing Two Models of Canine Ventricular Electrophysiology," *Am. J. Physiology - Heart and Circulatory Physiology*, vol. 292, pp. H43-H55, 2007.

[8] E.M. Cherry and F.H. Fenton, "Visualization of Spiral and Scroll Waves in Simulated and Experimental Cardiac Tissue," *New J. Physics*, vol. 10, p. 125016, 2008.

[9] P. Comtois and A. Vinet, "Curvature Effects on Activation Speed and Repolarization in an Ionic Model of Cardiac Myocytes," *Physical Rev. E*, vol. 60, no. 4, pp. 4619-4628, Oct. 1999.

[10] V.G. Fast and A. Kléber, "Role of Wavefront Curvature in Propagation of Cardiac Impulse," *J. Cardiovascular Research*, vol. 33, no. 2, pp. 258-271, Feb. 1997.

[11] F.H. Fenton and E.M. Cherry, "Models of Cardiac Cell," *Scholarpedia*, vol. 3, p. 1868, 2008.

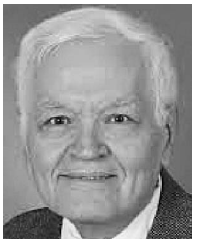
[12] F.H. Fenton, E.M. Cherry, H.M. Hastings, and S.J. Evans, "Multiple Mechanisms of Spiral Wave Breakup in a Model of Cardiac Electrical Activity," *Chaos*, vol. 12, no. 3, pp. 852-892, Sept. 2002.

[13] S.N. Flaim, W.R. Giles, and A.D. McCulloch, "Contributions of Sustained I_{Na} and I_{Kv43} to Transmural Heterogeneity of Early Repolarization and Arrhythmogenesis in Canine Left Ventricular Myocytes," *Am. J. Physiology - Heart and Circulatory Physiology,* vol. 291, pp. H2617-H2629, 2006.
[14] R. Grosu, G. Batt, F. Fenton, J. Glimm, C.L. Guernic, S. Smolka, and E. Bartocci, "From Cardiac Cells to Genetic Regulatory Networks," *Proc. 23rd Int'l Conf. Computer Aided Verification (CAV '11),* July 2011.
[15] V. Iyer, R. Mazhari, and R.L. Winslow, "A Computational Model of the Human Left-Ventricular Epicardial Myocytes," *Biophysical J.,* vol. 87, no. 3, pp. 1507-1525, 2004.
[16] M. Kay and R. Gray, "Measuring Curvature and Velocity Vector Fields for Waves of Cardiac Excitation in 2-d Media," *IEEE Trans. Biomedical Eng.,* vol. 52, no. 1, pp. 50-63, Jan. 2005.
[17] D.M. Khan, "Cubic Bézier Least Square Fitting," *Matlab File Exchange,* July 2009.
[18] A. Kléber, "The Fibrillating Atrial Myocardium. What Can the Detection of Wave Breaks Tell Us?" *J. Cardiovascular Research,* vol. 48, no. 2, pp. 181-184, Aug. 2000.
[19] D. Lloyd-Jones, R.J. Adams, T.M. Brown, M. Carnethon, S. Dai, G.D. Simone, T.B. Ferguson, E. Ford, K. Furie, C. Gillespie, A. Go, K. Greenlund, N. Haase, S. Hailpern, P.M. Ho, V. Howard, B. Kissela, S. Kittner, D. Lackland, L. Lisabeth, A. Marelli, M.M. McDermott, J. Meigs, D. Mozaffarian, M. Mussolino, G. Nichol, V.L. Roger, W. Rosamond, R. Sacco, P. Sorlie, R. Stafford, T. Thom, S. Wasserthiel-Smoller, N.D. Wong, and J. Wylie-Rosett, "Heart Disease and Stroke Statistics 2010 Update: A Report from the American Heart Association," *Circulation,* vol. 121, pp. e46-e215, Dec. 2009.
[20] C.H. Luo and Y. Rudy, "A Dynamic Model of the Cardiac Ventricular Action Potential. I. Simulations of Ionic Currents and Concentration Changes," *Circulation Research,* vol. 74, no. 6, pp. 1071-1096, June 1994.
[21] MATLAB, "Algorithm Isocontour," http://www.mathworks.com/matlabcentral/fileexchange/30525-isocontour, 2012.
[22] MATLAB, "Symbolic Math Toolbox," http://www.mathworks.com/help/toolbox/symbolic, 2012.
[23] L. Priebe and D.J. Beuckelmann, "Simulation Study of Cellular Electric Properties in Heart Failure," *Circulation Research,* vol. 82, pp. 1206-1223, 1998.
[24] D.F. Rogers and J.A. Adams, *Mathematical Elements of Computer Graphics.* McGraw-Hill, 1989.
[25] S.H. Strogatz, *Nonlinear Dynamics And Chaos: With Applications to Physics, Biology, Chemistry, and Enginnering (Studies in Nonlinearity).* Westview Press, 2001.
[26] J.V. Tranquillo, N. Badie, C.S. Henriquez, and N. Bursac, "Collision-Based Spiral Acceleration in Cardiac Media: Roles of Wavefron Curvature and Excitable Gap," *Biophysical J.,* vol. 98, no. 7, pp. 1119-1128, Apr. 2010.
[27] K.H.T. Tusscher, D. Noble, P.J. Noble, and A.V. Panfilov, "A Model for Human Ventricular Tissue," *Am. J. Physiology,* vol. 286, pp. H1573-H1589, 2004.
[28] A.T. Winfree, "Heart Muscle as a Reaction - Diffusion Medium: The Roles of Electric Potential Diffusion, Activation Front Curvature, and Anisotropy," *Int'l J. Bifurcation and Chaos,* vol. 7, no. 3, pp. 487-526, Mar. 1997.

Abhishek Murthy received the BE degree in computer engineering from the University of Mumbai, India, the MS degree in computer science from the University at Buffalo, and is currently working toward the PhD degree at Stony Brook University with Prof. Radu Grosu and Prof. Scott Smolka. He is a research assistant at the US National Science Foundations (NSF) working on the expeditions project on computational modeling and analysis of complex systems (CMACS). He is a student member of the IEEE and the IEEE Computer Society.

Ezio Bartocci received the BS degree in computer science, the MS degree in bioinformatics, and the PhD degree in information sciences and complex systems from the University of Camerino, Camerino, Italy, in 2002, 2005, and 2009, respectively. From 2010 to 2012, he was a postdoctoral researcher in the Department of Applied Math and Statistics at the State University of New York at Stony Brook. He is currently an assistant professor in the Department of Computer Engineering at Vienna University of Technology.

Flavio H. Fenton received the BS degree in theoretical physics from the Universidad Nacional Autonoma de Mexico, and the MS and PhD degrees in physics from Northeastern University. He is currently a research associate in the Department of Biomedical Sciences at Cornell University. He is a principal investigator at the US National Science Foundations (NSF) working on the expeditions project on computational modeling and analysis of complex systems (CMACS).



James Glimm received the PhD degree from Columbia University in 1959. He has been noted for contributions to C*-algebras, quantum field theory, partial differential equations, fluid dynamics, scientific computing, and the modeling of petroleum reservoirs. Together with Arthur Jaffe, he founded a subject called constructive quantum field theory. He was elected to the National Academy of Sciences in 1984. He won the National Medal of Science in 2002. Currently, he is a full professor in the Department of Applied Mathematics and Statistics at Stony Brook University.

Richard A. Gray received the BS degree in chemical engineering from Bucknell University, and the MS and PhD degrees from the University of Virginia. He served as an associate professor in the University of Alabama's Department of Biomedical Engineering. He is currently a scientist with the Food and Drug Administration. He is a member of the IEEE.

Elizabeth M. Cherry received the BS degree in mathematics from Georgetown University and the PhD degree in computer science from Duke University. She is currently an assistant professor in the School of Mathematical Sciences at the Rochester Institute of Technology. Her research interests include mathematical modeling and computer simulation of cardiac electrophysiology, numerical methods for excitable media, and nonlinear dynamics.

Scott A. Smolka received the AB and AM degrees in mathematics from Boston University and the PhD degree in computer science from Brown University. He is a professor in the Department of Computer Science at the State University of New York, Stony Brook. He is also the president and cofounder of Reactive Systems, Inc. He has been on the faculty at Stony Brook University since 1982. His work focuses on process algebra, model checking, and runtime verification, with application to probabilistic systems and excitable cells (cardiac cells). He is perhaps best known for the algorithm he and Paris Kanellakis invented for deciding bisimulation. He is a member of the IEEE and the IEEE Computer Society.

Radu Grosu received the Dr.rer.nat. degree in computer science from the Technical University of Muenchen. He is a professor and chair of the Dependable-Systems Group in the Faculty of Informatics at the Vienna University of Technology, and a research professor in the Computer Science Department at the Stony Brook University. His research interests include modeling, analysis, and control of cyberphysical and biological systems and his application focus includes green operating systems, mobile ad hoc networks, automotive systems, the Mars rover, cardiac-cell networks, and genetic regulatory networks. He is the recipient of the US National Science Foundation Career Award, the State University of New York Research Foundation Promising Inventor Award, and the ACM Service Award, and a member of the IFIP WG 2.2. Before receiving his appointment at the Vienna University of Technology, he was an associate professor in the Computer Science Department at the Stony Brook University, where he codirected the Concurrent-Systems Laboratory and cofounded the Systems-Biology Laboratory. He was a research associate in the Computer Science Department at the University of Pennsylvania.