Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

SIMULATION OF NONLINEAR DISTRIBUTED PARAMETER SYSTEMS ON THE CONNECTION MACHINE

Boris Y. Kogan
Walter J. Karplus
Alex T. Pang

June 1989
CSD-890026

Simulation of Nonlinear Distributed Parameter Systems on the Connection Machine

Boris Y. Kogan
Walter J. Karplus
Alex T. Pang

Computer Science Department
University of California, Los Angeles

Abstract

The recently introduced Connection Machine, CM-2, is a multiprocessor with massive parallelism. Fully expanded, the CM-2 consists of 64K processing elements connected in a hypercube topology. Each processing element contains a single-bit arithmetic and logic unit as well as 64K bits of local memory. Operation is in the SIMD mode with a single control unit controlling all the processors. In this paper the application of a quarter CM-2 (16K processing elements) to the simulation of excitable media is described. Excitable media are distributed parameter systems, characterized by nonlinear partial differential equations, containing distributed sources of energy. Excitable media arise in biological systems (e.g. heart muscles), chemical processes and a variety of other application areas. The implementation of a mathematical model for the heart muscle on the CM-2 permitted the generation of very interesting computational results.

Key Words: Multiprocessors, Massively Parallel Processors, Connection Machine, Excitable Media, Heart Muscle Simulation.

1. Introduction

The requirement for larger and more powerful digital computers for the solution of challenging scientific computing problems has lead to the introduction of a wide variety of innovative high performance systems. The supercomputers and mini-supercomputers now on the market achieve a very high computational throughput with the aid of architectures manifesting a variety of combinations of pipelining (vector processing) and parallelism. The supercomputers, such as CRAY Y-MP, CRAY2, ETA-10, and competing systems introduced by NEC, Hitachi, and Fujitsu, as well as mini-supercomputers, such as the Convex-240 and the FPS

M64, provide a number of concurrently operating vector pipelines and scalar processing units. Shared memory multiprocessors, including those marketed by Alliant Computer Corp, Encore Computer Corp, Elxsi, and Flexible Computer Corp, provide up to 32 processing elements (each containing its own control unit and therefore operating in the MIMD mode) all of which have direct access to a single large shared memory. Yet another class of high performance computers, including those marketed by Intel Scientific Computers, NCUBE and Ametek, provide up to 128 concurrently operating processing elements, each with its own control unit. Each of these approaches to high performance have a number of advantages and disadvantages and entail judicious tradeoffs as described by Hwang (1) and Karplus (2, 3).

A very recent entrant into the high performance scientific computing field was introduced by Thinking Machines Corp and is called the Connection Machine. The first version of the Connection Machine, CM-1, became available in 1986, while the second model, the CM-2, appeared in 1987. These systems contain from 8K to 64K processing elements. This makes it possible to envision the implementation of simulation models heretofore considered impractical and the development of new and more powerful algorithms and computational procedures.

It is the purpose of this paper to describe the application of CM-2 to the simulation of an interesting and important class of distributed parameter systems – excitable media, which arise in biology, chemical processes, and a number of other areas. To this end, the architecture of CM-2 and its software system are first described. This is followed by an introduction to excitable media and their mathematical models. Some details of the implementation of a specific model and the simulation results are then provided.

2. The Connection Machine

The Connection Machine, as described by Hillis (4) and Tucker and Robertson (5) differs from earlier SIMD systems principally in the complexity of the network interconnecting the processing elements.

2.1 System Architecture

A block diagram of the fully-expanded Connection Machine is shown in Fig. 1. One or more front-end machines, such as a Symbolics, Sun or Vax workstation, serve as the external interface. These front-end computers connect with up to four machine quadrants through the Nexus 4 x 4 crossbar switch. Each quadrant contains 1024 monolithic integrated devices, each of which in turn contains 16 processing elements. Therefore, each quadrant has 16K processing elements. Each processing element includes an arithmetic and logic unit (ALU) and 64K bits of bit-addressable random access memory. In addition, each processing element contains four 1-bit flag registers, an I/O interface, and share an optional floating-point accelerator (among 32 processing elements). This structure is illustrated in Fig. 2. The ALU is of a very simple design

and operates in a bit-serial fashion, which requires 750 nanoseconds per bit plus instruction decoding and overhead. The greater the required precision, the lower the computational throughput. For example, a 32 bit addition requires slightly over 24 microseconds. Note, that the processing elements do not contain control units. They are controlled by the sequencers which convert a stream of high level instructions and arguments into microcode instructions (*nanoinstructions*) that control the timing and operation of the processing elements. All processing elements in a quadrant must execute the same instruction in the familiar single instruction-multiple data stream (SIMD) mode.

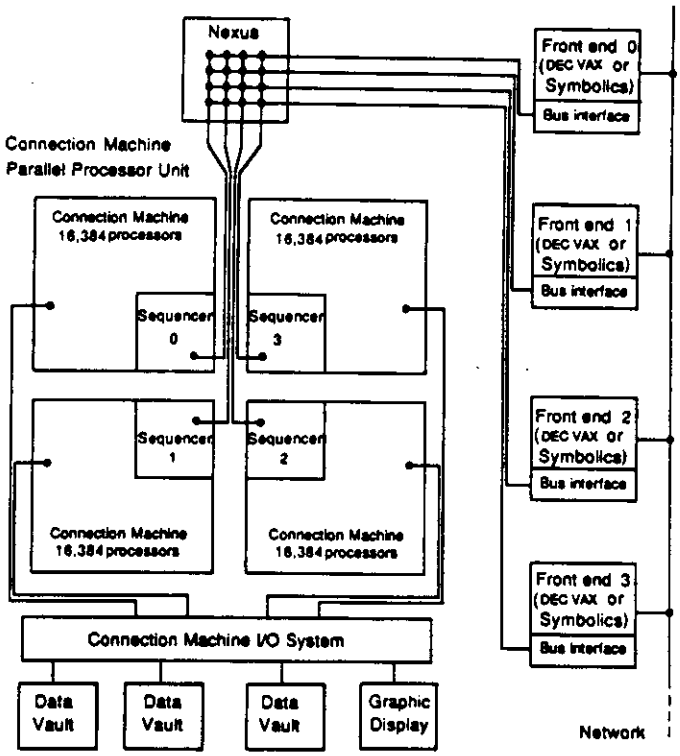

Figure 1. General Organization of the Connection Machine

There is provision for two types of interprocessor communications. The NEWS (for north, east, west and south) grid is a two dimensional grid permitting each processor to communicate directly with its four nearest neighbors. In addition, all of the integrated devices (each containing 16 processing elements) are interconnected by a router network which has the topology of a ten dimensional boolean hypercube (for a quarter machine). This network provides for efficient, pipelined message switching so as to assure optimal routes for messages when there is heavy traffic.

Efficient I/O is achieved by providing every quadrant with two I/O channels. A channel may be connected either to the frame buffer of a high resolution graphics display or a general I/O controller. Data is transferred directly and in parallel between the I/O devices and the processing elements at a rate of 40 megabytes per second per I/O controller. Transfers between the data processors and the frame buffers can be effected at a rate of one gigabits per second, enough to produce full colored, 24 bit per pixel images at a rate of over 30 frames per second for a 1280×1024 display.

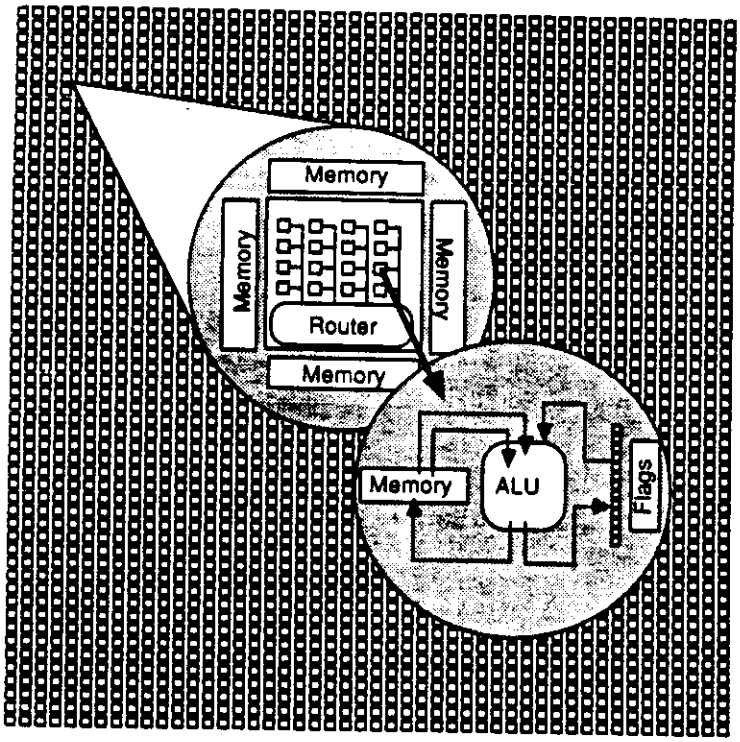

Figure 2. Exploded View of the Connection Machine

2.2 Software

Program development for the CM-2 is supported by three high level languages: *Lisp, CM Fortran and C*. Software is written in the front-end computer and compiled into the *Paris* (parallel instruction set) language that can be issued to the parallel processing units. It is the lowest level protocol by which the front-end computer directs the actions of the CM-2 processors. *Paris* instructions are sent to a micro-controller which expands them into a series *nanoinstructions*, of which there may be from one to several thousand per *Paris* instruction, and then broadcasts these sequentially to all the processors.

Data level parallelism uses a single control sequence or program, where code is executed sequentially. All programs are stored in the front-end computer. Note that the processor memory contains only data and immediate computations, and does not contain code for execution. Thus any fetches from local memory are simply to load the ALU for a boolean operation. Each processor memory is split into two parts: stack and data. In keeping with the SIMD nature of the machine, a single system wide stack pointer and a single system wide stack limit register are used. Each processor has a 1-bit context flag that determines whether or not a processor will respond to global instruction or not. As each *nanoinstruction* is executed, the data from all the memory location pointed to by the stack pointer are fetched from all the processors that have their context flag bit on. Setting the context flag provides basic conditional control equivalent to if-then-else statements.

In situations where the number of physical processors is not enough to adequately meet the degree of data level parallelism, the Connection Machine can support virtual processors. When the Connection Machine is initialized for a particular application, the number of virtual processors required by the application may be specified. For example, if an application needs a million (2^{20}) virtual processors, with 16K (2^{14}) of physical processors, each processor would support 64 virtual processors. Also, the 64K (2^{16}) bits of physical memory would have to be divided among the 64 (2^{6}) virtual processors. Thus, each virtual processor has about 1K bits of memory. The execution speed of a virtual processor would also appear to be roughly 1/64 that of a physical processor.

3. Excitable Media

This paper is concerned with the application of a single-quadrant CM-2 to the simulation of a class of systems characterized by complex nonlinear mathematical models and which exhibits spectacular, dynamic behavior. So-called self-organizing structures appear in a wide variety of physical, chemical and biological system. They are characterized by imbalances and disequilibrium which result in the appearance of evolving wave fronts propagating through the system in characteristic, though very complex, patterns. These self-organizing processes take place in excitable media. Madore and Freedman (6) describe this phenomenon as it is observed

in relatively simple chemical reactions (Belousof-Zhabotinskii reaction), the growth of communities of a particular type of amoeba, as well as the spiral structures seen in many galaxies. An important practical application of the concept of excitable media involves the study and simulation of the electrical wave propagation along heart muscles to determine the electrophysiological nature and causes of heart fibrillation.

Excitable media belongs to a particular class of nonlinear distributed systems whose main characteristics are:

1. The system can be regarded as a set of small interacting parts distributed in space. The interaction between different parts is through diffusion.

2. Excitable media are *active* distributed systems. Each component possess nonlinear dynamic properties and its own internal energy source.

3. In response to some external stimulus, the super-threshold excitation may change the steady state response, generating a single pulse or a train of pulses depending on the nonlinear properties of the medium.

4. The shape of generated pulses and the characteristic of pulse activity do not depend on the form of the external forcing function.

5. Each part of the system that has been excited beyond a certain threshold goes through a refractory period during which repeated external disturbance, no matter how strong, cannot produce another excitation.

In homogeneous excitable media, the following wave phenomena can be observed:

1. Propagation of a traveling excitation wave without attenuation.

2. Circulation of excitation wave that leads to the formation of spiral waves.

3. Generation of concentric waves by autonomous leading centers.

4. The occurrence of dissipative structures – space inhomogeneous and time stationary distribution of states among the different parts of the excitable media.

An important feature of the wave phenomena mentioned above, is that their stationary regimes do not depend on initial conditions. By analogy with auto-oscillations in lumped dynamical systems, these wave processes are called *auto-wave processes*.

Processes in excitable media can be represented mathematically (7) as

$$\frac{\partial E_i}{\partial t} = \nabla(D_i \nabla E_i) + G_i(\nabla E_i, E_i) + F_i(x,t) \qquad i = 1,2,...,n \tag{1}$$

with specified initial and boundary conditions. Here E_i are the state variables, G_i are nonlinear functions of E_i and perhaps ∇E_i. D_i are diffusion coefficients and $F_i(x,t)$ are external disturbances. The gradient operator ∇ is defined as:

$$\nabla = \frac{\partial}{\partial x_1} i + \frac{\partial}{\partial x_2} j + \frac{\partial}{\partial x_3} k + ...$$

The mathematical models characterizing specific systems of interest tend to be very complex and often include more than ten components with a wide range of time constants. The greatly differing time constants introduce stiffness and difficulties in carrying out numerical integration. In addition, the complex character of the nonlinearities in G_i and the necessity to regard the behavior of a large number of interacting parts which form the excitable medium (specially in with three dimensional cases) make this problem very difficult. Several attempts have been made to simulate such a system. A two dimensional excitable media problem was solved many years ago by Kogan et. al. (8) using a hybrid computer; and three dimensional problems were attempted by Nandapurkar and Winfree (9) using a supercomputer. These investigations showed that such simulations even with supercomputers using very simplified models are very costly and time consuming.

4. Simplified Mathematical Model

In order to investigate the suitability of the Connection Machine for the simulation of excitable media, a simplified mathematical model was selected in order to facilitate the comparison of the results obtained with CM-2 with those available in the literature. FitzHugh (10) and Ivanitsky et. al. (11) have shown that Eqn. 1 can be reduced to a system of equations of the form

$$c\frac{\partial E}{\partial t} = D\left[\frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2}\right] - I_{fast}(E) - I - I_{stimulus} \tag{2a}$$

$$\frac{\partial I}{\partial t} = \frac{I_{slow}(E) - I}{\tau_{slow}} \tag{2b}$$

with given initial conditions $I(0,x,y) = E(0,x,y) = 0$ and boundary conditions $\frac{\partial E}{\partial x} = 0$ and $\frac{\partial E}{\partial y} = 0$.

Eqn. 2a describes the fast process of wave front formation and propagation, while Eqn. 2b describes the slower recovery or refractory process of the media. Where the heart muscle is the excitable medium, the variable E denotes a displacement of the membrane potential between the interior and exterior of the cell. I is the generalized ionic outward current. $I_{stimulus}$ is the external forcing function. $I_{slow}(E)$ denotes the steady state dependence of the slow outward current I on the membrane potential E. c is the membrane capacitance, and the parameter D is a diffusion coefficient

$$D = \frac{r}{2R_{ic}} 10^{-4} \ [mho]$$

where r $[micron]$ is the fiber radius and R_{ic} $[ohm \cdot cm]$ are the specific resistances of the intracellular media.

For a heart muscle, $r = 10$ $[micron]$, $R_{ic} = 150$ $[ohm \cdot cm]$. So, $D = 3.33 \times 10^{-3}$ $[m\ mho]$. Likewise, the membrane capacitance for heart muscle $c = 12$ $[\mu F / cm^2]$. $\tau_{slow} = 100$ $[msec]$. The units of the other variables are t $[msec]$, I $[\mu A / cm^2]$, E $[mV]$, x and y $[cm]$.

Eqn. 2a and 2b can be reduced to the non-dimensional form

$$\frac{\partial \bar{E}}{\partial \bar{t}} = \left[\frac{\partial^2 \bar{E}}{\partial \bar{x}^2} + \frac{\partial^2 \bar{E}}{\partial \bar{y}^2} \right] - \bar{I}_{fast}(\bar{E}) - \bar{I} - \bar{I}_{stimulus} \tag{3a}$$

$$\frac{\partial \bar{I}}{\partial \bar{t}} = \varepsilon(\bar{E}) \left[\bar{I}_{slow}(\bar{E}) - \bar{I} \right] \tag{3b}$$

The same initial and boundary conditions apply to the transformed variables. Note that the transformed variables are identified by bars. Here, $\bar{t} = \frac{t}{\tau_{fast}}$, $\bar{x} = \frac{x}{\sqrt{D / g_{fast}}}$, $\bar{y} = \frac{y}{\sqrt{D / g_{fast}}}$, $\bar{E} = \frac{E}{E_{\max}}$, $\bar{I} = \frac{I}{I_{\max}}$.

g_{fast} is the conductivity for the fast ion current in $[m\ mho / cm^2]$, $E_{\max}$ is the maximum value of membrane potential. $I_{\max}$ is the maximal current. $\tau_{fast} = \frac{c}{g_{fast}}$.

The model described by Eqn. 3a and 3b is termed the basic model. Pertsov et. al. (12) showed that the model can be further simplified by replacing the complex nonlinear functions $\bar{I}_{fast}(\bar{E})$, $\bar{I}_{slow}(\bar{E})$ and $\varepsilon(\bar{E})$ by their piecewise linear approximation as shown in Fig. 3. Therefore,

$$\bar{I}_{fast} = \begin{cases} G_r\bar{E} & \bar{E} < \bar{E}_1 \\ -G_f\bar{E} + A & \bar{E}_1 \le \bar{E} \le \bar{E}_2 \\ G_r(\bar{E} - 1) & \bar{E} > \bar{E}_2 \end{cases} \tag{4}$$

$$\bar{E}_1 = \frac{A}{G_r + G_f}; \quad \bar{E}_2 = \frac{G_r + A}{G_r + G_f}; \quad \bar{E}_{th} = \frac{A}{G_f}$$

$$\varepsilon(\bar{E}) = \begin{cases} 0.5 & \bar{E} \le 0.01 \\ 0.06 & 0.01 < \bar{E} < 0.95 \\ 0.5 & \bar{E} \ge 0.95 \end{cases}$$

$$\bar{I}_{slow}(\bar{E}) = G_s\bar{E}$$

$\bar{E}_{th}$ is the threshold potential (see Fig. 4). G_s is a constant coefficient.

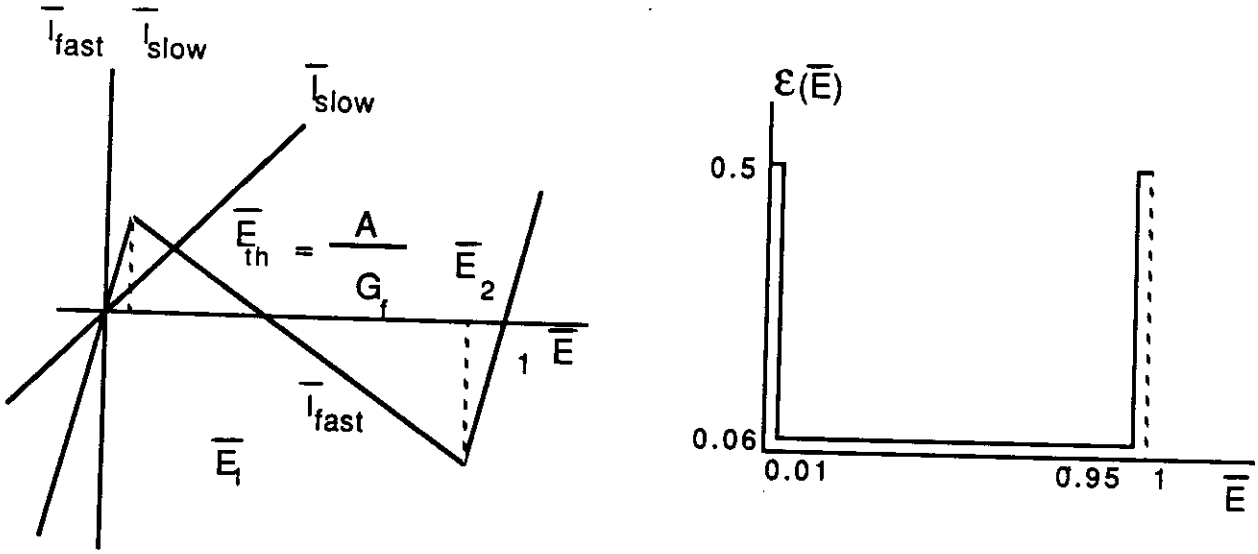

Figure 3: Nonlinear Functions

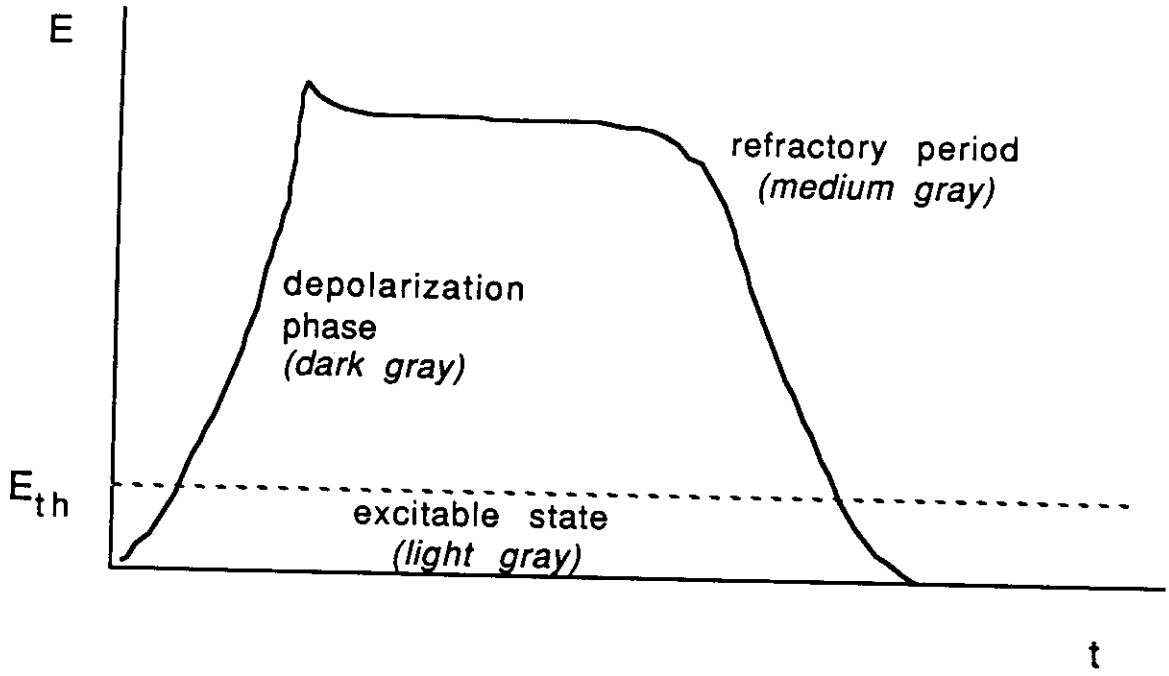

Figure 4. Action Potential – Excitation Pulse

5. Finite Difference Approximation

The objective of the simulation is to find the distribution $E = f(x,y,t)$ in the heart muscle using the simplified mathematical model. Discretization of space coordinates and finite difference approximation reduce the mathematical model described by Eqn. 3a and 3b to a set of ordinary differential equations

$$\frac{d\bar{E}_{ij}}{d\bar{t}} = \bar{I}_{ext_{ij}} - \bar{I}_{fast}(\bar{E}_{ij}) - \bar{I}_{ij} - \bar{I}_{stimulus_{ij}} \tag{5a}$$

$$\frac{d\bar{I}_{ij}}{d\bar{t}} = \varepsilon(\bar{E}_{ij}) \left[I_{slow}(\bar{E}_{ij}) - \bar{I}_{ij} \right] \tag{5b}$$

$$\bar{I}_{ext_{ij}} = \frac{\bar{E}_{i-1,j}(\bar{t}) - 2\bar{E}_{ij}(\bar{t}) + \bar{E}_{i+1,j}(\bar{t})}{\Delta\bar{x}^2} + \frac{\bar{E}_{i,j-1}(\bar{t}) - 2\bar{E}_{ij}(\bar{t}) + \bar{E}_{i,j+1}(\bar{t})}{\Delta\bar{y}^2} \tag{5c}$$

The initial distributions of $\bar{E}_{ij}(0)$ and $\bar{I}_{ij}(0)$ are specified. The boundary conditions for the boundary nodes are $\frac{\partial\bar{E}}{\partial\bar{x}} = 0$ and $\frac{\partial\bar{E}}{\partial\bar{y}} = 0$. The functions $\bar{I}_{fast}(\bar{E})$, $\bar{I}_{slow}(\bar{E})$ and $\varepsilon(\bar{E})$ are the same as in Eqn. 4.

In order to take advantage of the data level parallelism in the CM-2, the method of lines is chosen. Eqn. 5a and 5b are computed for each grid point during each integration time step. A slight variation of the method of lines is used to reduce the amount of communication between neighboring grid points. This is achieved by holding off the evaluation of Eqn. 5c to integral multiples of the integration time step. The choice of this communication interval is such that the interval is small enough so that $\bar{I}_{ext}$ can be assumed to be constant over that interval. This is explained in more detail in the next section.

6. Implementation on the CM-2

The UCLA quarter CM-2 Machine with 16K processors was used for two dimensional and three dimensional simulations of normal and abnormal wave propagations.

It was shown by Moe et. al. (13) that many interesting phenomena (particularly spiral waves) can be observed in excitable media if the number of nodes exceeds 1000. In our simulation the full power of the available CM-2 processors was used. Therefore, a grid of 128×128 or 16,384 nodes was chosen for the 2D case, and $128 \times 128 \times 22$ was chosen for 3D. In the latter case, the virtual to physical processor ratio was 22. This means that each physical processor simulated 22 virtual processors, enabling us to simulate 20 layers of excitable tissue. The two other layers were used for boundary conditions.

The parameter values in Eqn. 4 were made the same as in previous simulation studies using serial computers (12). That is, $G_s = 1, G_f = 0.735, G_r = 30, \frac{A}{G_f} = 0.16$. It was also observed that simplifying the $\varepsilon(\bar{E})$ function as follows did not significantly alter the refractory part of the wave:

$$\varepsilon(\bar{E}) = \begin{cases} 0.5 & \bar{E} \le 0.01 \\ 0.03 & 0.01 < \bar{E} \le 1 \end{cases}$$

Note that for $\bar{E} > 0.01$, the choice of $\varepsilon(\bar{E}) = 0.03$ or 0.06 (as in Eqn. 4 and Fig. 3) only serves to change the duration of the action potential. Choosing $\varepsilon(\bar{E})$ to be 0.03, the conversion between the non-dimensional form of space and time are 0.05 [*cm/space unit*], and 3 [*msec/time unit*] respectively. The space step was chosen to be $\Delta\bar{x} = \Delta\bar{y} = 0.85$ and the integration step $\Delta\bar{t} = 0.03$. For a 128×128 grid, this corresponds to a tissue of 5.44×5.44 [cm^2]; and a simulation time of 500 *time units* corresponds to 1.5 *seconds*. A second order Runge Kutta formula, RK-2, was used for the simulations.

The time of $I_{ext_{ij}}$ exchanges between the grid elements was chosen to be $2\Delta\bar{t}$ for the normal propagation case and $3\Delta\bar{t}$ for all the others. The choice of this time is dictated by considerations of stability and accuracy of the numerical calculations. The time domain for the simulation was T = 250 time units for normal propagation and T = 500 time units for all the others. This means that for the runs with 500 time units, all 16,384 nodes in the system performed 16,666 pairs of integration steps (each requiring two function evaluations for RK-2) and 5,555 sets of four nearest neighbor communications. Each finite difference node point was represented by a CM-2 processor. All computations were carried out in a bit-serial fashion using a 32-bit floating point format. The program was written in C* and compiled on a Vax front-end computer. The simulation environment is illustrated in Fig. 5. The following is an outline of the algorithm:

1. *Serial code.* Read in the system description file. This file contains information regarding the user modifiable parameters such as initial conditions, stimulus, abnormalities in the tissue, etc.

2. *Parallel code.* This section of the code applies to each individual processing element in the CM-2.

   a. Initialize the state variables. $\bar{I}_{ext_{ij}}$ is initially set to 0.

   b. Evaluate Eqn. 5a and 5b.

   c. Carry out nearest neighbor communication in order to evaluate Eqn. 5c. Note that this step may be carried out after an integral multiple of step (b).

   d. Display $\bar{E}$. Again, this step may be performed after an integral multiple of step (b).

3. *Serial code.* Advance the global simulation time by $\Delta\bar{t}$. Go to step (b) unless the total simulation time has been exceeded.

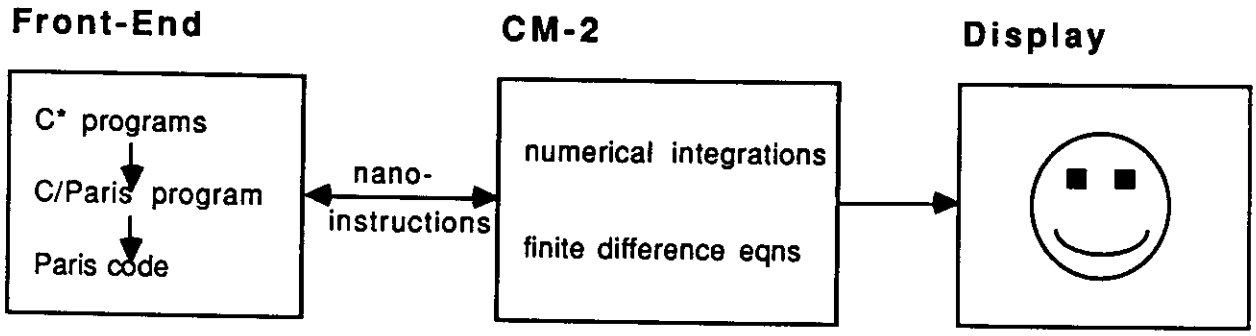

Figure 5. Implementation Environment

7. Simulation Results

The results obtained in the computer simulation are presented in Fig. 6 to 9. In interpreting the pictures, it should be recognized that each node of the 128×128 grid represents a cell whose behavior is shown in Fig. 4. The different phases of the cells over time are easily extracted by using the following coloring scheme: referring to Fig. 4., all cells whose excitation level is below $\bar{E}_{th}$ are colored blue; those cells which are on the depolarization phase, shown as the steep rise in potential, are colored red; those on the refractory or recovery phase are colored green. Additionally, cells which are temporarily *dead* or unexcitable are colored pink; while borders and holes are colored black. Note that holes behave the same way as boundary points; that is, there is no flux across the boundaries. Finally, those cells that are colored yellow are tracking the movement of the *inflection point* where the forward moving wave front (head of the wave) and the retreating wave (tail of the wave) coincides.

The series of pictures in Fig. 6 shows the normal wave propagation when a small area in the upper left corner of the tissue is stimulated. The process of excitation propagation is represented in the form of a series of successive positions of the excited domain, that is of groups of points of the heart whose potential exceeds at the given instant a certain threshold value $\bar{E}_{th}$.

The potential of the points on the boundary of the excited area is, by definition, equal to $\bar{E}_{th}$. When an excitation pulse is generated, this value is attained twice: in the initial stage of excitation ($\partial \bar{E} \,/\, \partial \bar{t} > 0$) and in the final phase ($\partial \bar{E} \,/\, \partial \bar{t} < 0$) (see Fig. 4). Accordingly, on the boundary of the excited area one can distinguish two segments: the first segment, called the wave front, consists of excited points and moves toward the unexcited area; the other segment of the boundary, called the tail of the excitation wave, consists of points which come out of the state of excitation and moves toward the excited area.

In the case of normal excitation wave propagation, each point of the excitable medium generates a pulse shifted in time by an amount depending on the distance from the source of the original excitation. The front and the tail of the wave succeed each other at a distance equal to the wave length $\bar{\lambda} = \bar{\theta}\bar{D}$, where $\bar{D}$ is the duration of the excitation pulse; that is the time during which the value of the potential exceeds $\bar{E}_{th}$. $\bar{\theta}$ is the speed of propagation.

To create the spiral waves shown in the Fig. 7 sequence, a temporarily dead tissue was introduced. As the wave reaches around the bottom of the dead tissue, the area under the dead tissue is suddenly made excitable again. Similar spiral waves have been noted in laboratory experiments with strips of heart tissue.

The double opposing spiral waves leading to concentric waves further out from the center, as depicted in the Fig. 8 sequence, are easy to produce. One simply moves the temporary dead tissue toward the interior of the tissue. This causes the wave front to break at both ends of the dead tissue creating the two centers of the resulting spiral waves. This phenomenon, although not seen in previous simulation experiments involving the heart model, proves the possibility of the presence of leading centers in heart muscle. It should also be pointed out that similar phenomena are commonly observed in chemical reactions such as the Belousof-Zhabotinskii reaction.

The last sequence of images in Fig. 9 shows a rotating wave around a hole located in the center of the medium. This was obtained by placing a temporarily dead zone bridging one boundary to the hole. Since the stimulus is applied to the interior of the medium, it would naturally result in a circular wave propagation away from the point of stimulus. However, since the dead tissue effectively eliminates half of this wave, the remaining wave simply goes around the hole. This phenomenon resembles the observations of similar electrical wave circulations around the entry of cava veins to the atrium.

A typical simulation run consisting of 500 time units of simulation time requires about 6.5 minutes of computing time on the quarter CM-2. This can broken down among the principal tasks as shown in Table 1.

numerical integrations	190 *sec*
$I_{ext_{ij}}$ calculations	185 *sec*
graphics display	15 *sec*
Total	390 *sec*

Table 1. Breakdown of Computational Task for the RK-2 Algorithm

Table 2 presents a rough comparison of the computing time requirements of the implementation described in this paper and those reported by other investigators. All three pure digital implementations employed the simplified FitzHugh-Nagumo equation (10) described by Eqn. 2a and 2b. Only Kogan et. al. (8) used a different, more complicated form of the equation (14).

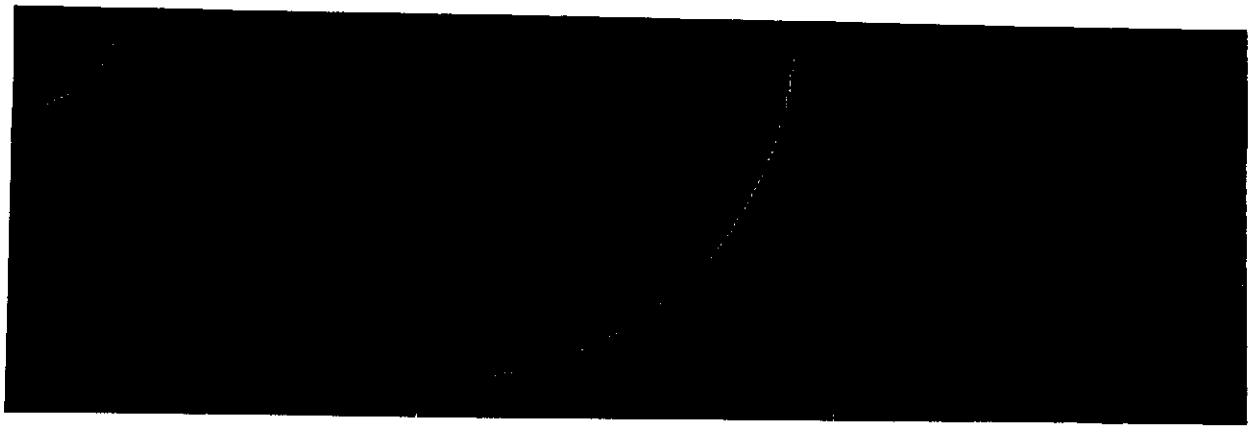

Figure 6. Normal Wave Propagation

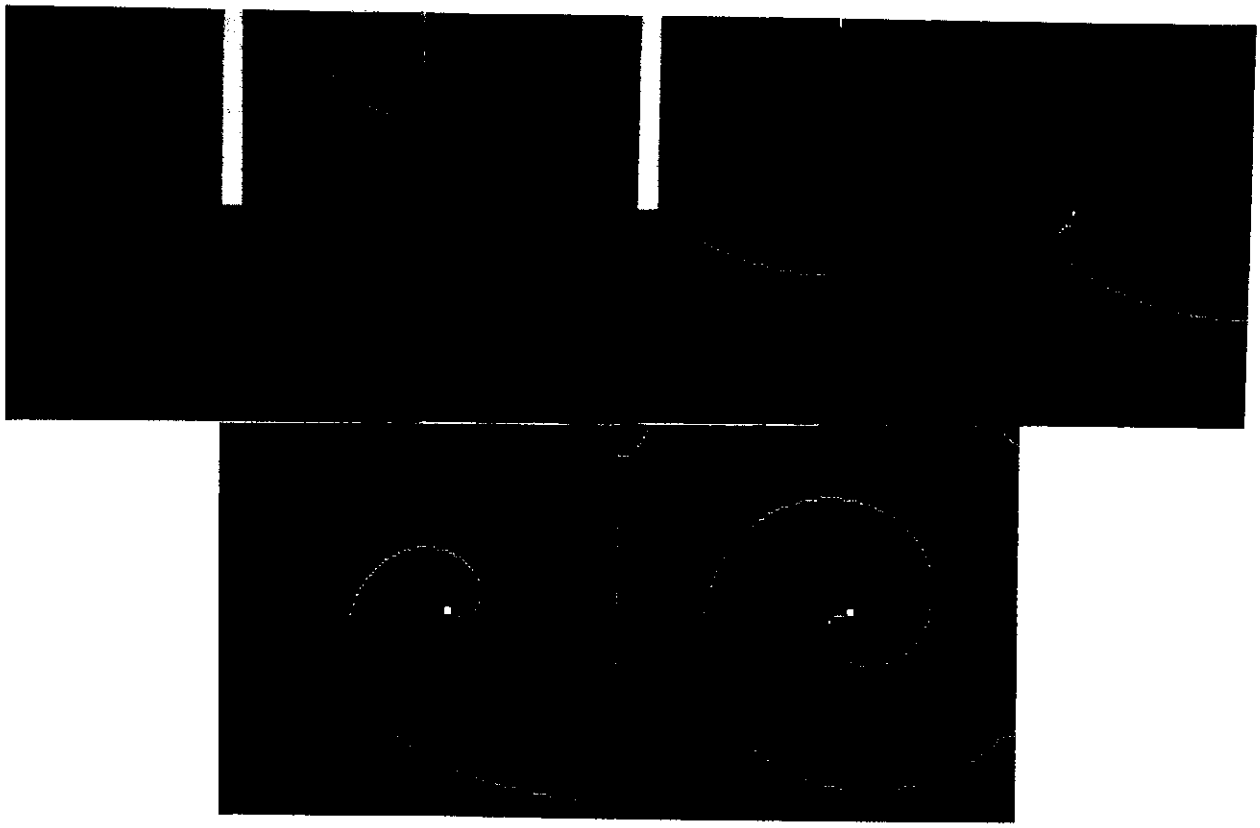

Figure 7. Spiral Wave Propagation

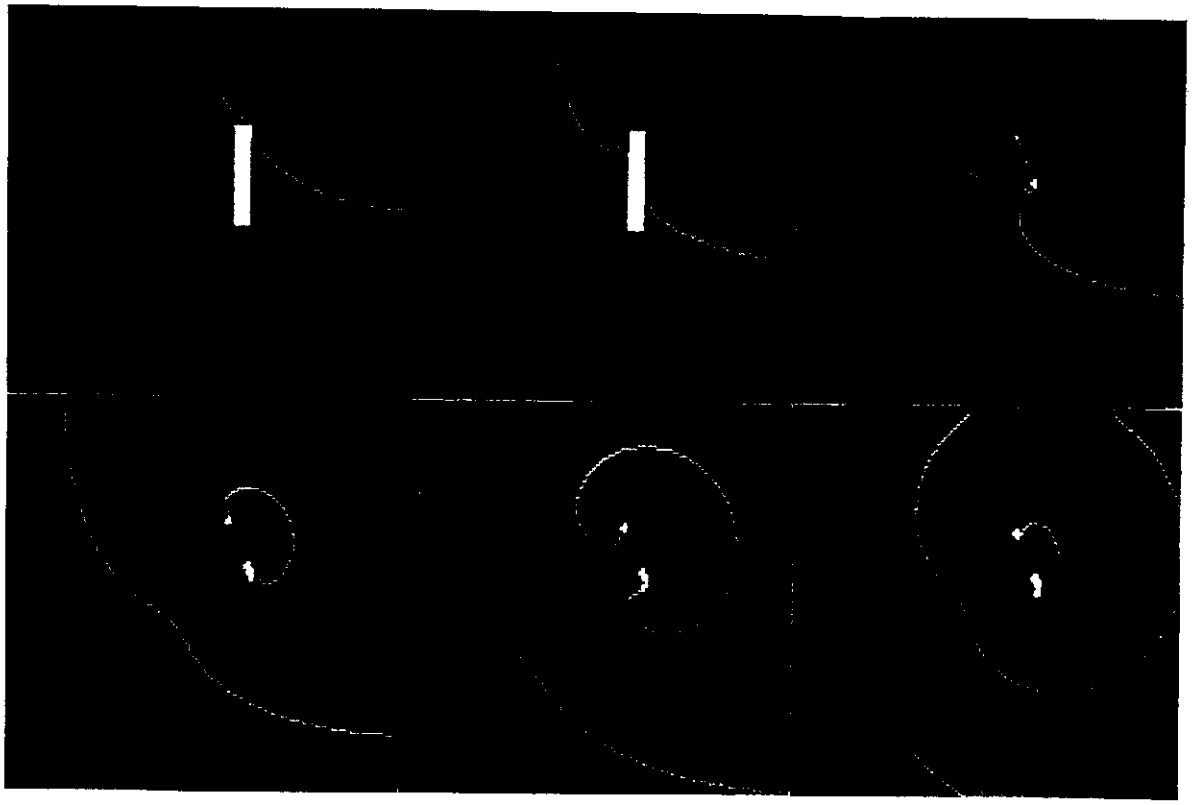

Figure 8. Double Spiral Wave Propagation

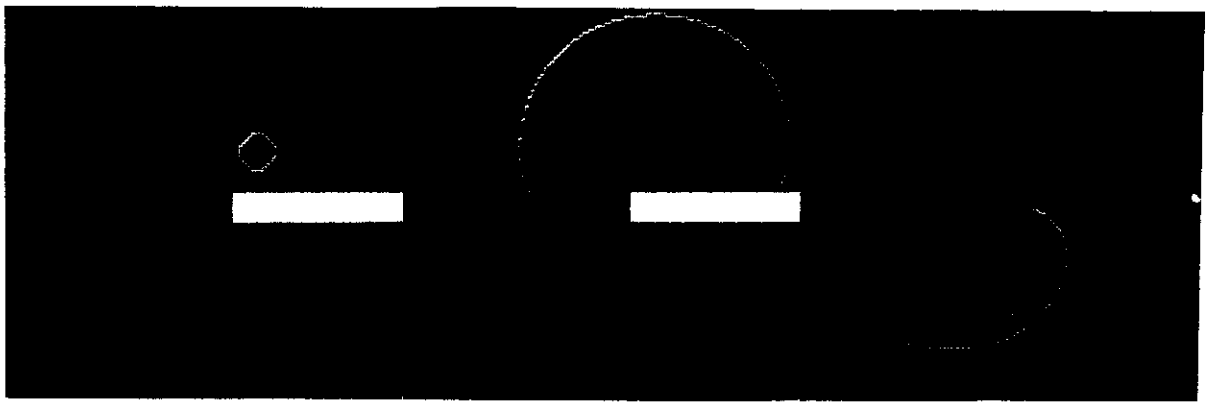

Figure 9. Revolving Wave Propagation

	Kogan, Zykov, Petrov (1980)	Pertsov, Ermakova, Panfilov (1984)	Nandapurkar, Winfree (1987)	Kogan, Karplus Pang (1989)
type of computer used	HCS-100 (hybrid)	ES 10/40 (IBM)	Cyber 205	CM-2 - 1/4
grid dimension	32×32	34×80	$41 \times 41 \times 41$	128×128
integration algorithm	continuous (analog)	Euler	Euler	RK-2
integration time step Δt		0.05	0.03	0.03
$I_{ext_{ij}}$ intervals (*time units*)	0.83	0.05	0.03	0.09 ($3\Delta t$)
Total simulated time (*time units*)	250	10	100	500
Total computer time (*sec*)	600	2400	80	390

Table 2. Comparison of Other Simulation Experiments

8. Conclusion

The results of the investigation presented in this paper suggest that massively parallel processors, such as the Connection Machine, constitute valuable and effective vehicles for the simulation of excitable media. The high degree of parallelism permitted us to investigate a larger grid size and in the process allowed us to observe double spiral waves rotating in opposite directions. The concentric waves resulting from the double spirals were not observed in previous numerical simulations of the heart muscle model.

The ease of use and the relatively short turn-around time of the CM-2 encourages us to investigate related areas with more vigor. For example, we have started an investigation of nonhomogeneous excitable media. Another step forward would be to transcend the over-simplified FitzHugh-Nagumo model and to pursue the simulation of more accurate and detailed models. As pointed out above, we have also implemented a three dimensional version of the FitzHugh-Nagumo model. The larger grid dimensions we are using are expected to permit us to generate interesting results that could not be obtained with the 41^3 grid used in (9). With the future availability of more and improved processing elements, massively parallel computing promises new results and new avenues for doing the numerical simulation of nonlinear distributed parameter systems in general, and for excitable media in particular.

9. Acknowledgement

Research in the use of massive parallelism for system simulation in the UCLA Computer Science Department is supported in part by the NASA/Dryden Research Center under Grant No. FCC 2-374, Project No. 18.

References

[1] Hwang, K. and D. DeGroot (Eds), "Parallel Processing for Supercomputers and Artificial Intelligence", McGraw Hill Book Company Inc., 1989.

[2] Karplus, W. J. (Ed), "Multiprocessors and Array Processors", Society for Computer Simulation, San Diego, California, Simulation Series Volume 18, No. 2 1987.

[3] Karplus, W. J. (Ed), Multiprocessors and Array Processors, Society for Computer Simulation, San Diego, California, Simulation Series, Volume 21 No. 1, 1989.

[4] Hillis, W. D., "The Connection Machine", MIT Press, Cambridge, Mass., 1985.

[5] Tucker, L. W. and Robertson, G. G., "Architecture and Applications of the Connection Machine", IEEE Computer, Volume 21, pp.26-38, August 1988.

[6] Madore, B. F. and Freedman, W. L., "Self-Organizing Structures", American Scientist, Volume 75, pp.252-259, May-June 1987.

[7] Haken, H. "Synergetics Nonequilibrium Phase Transitions and Self-organization in Physics, Chemistry and Biology" (second edition), Springer-Verlag Berlin-Heidelberg-New York, 1978.

[8] Kogan, B., Zykov V. S. and Petrov, A. A., "Hybrid Computer Simulation Of Excitable Media", Proceedings of the Ninth IMACS World Congress, North-Holland, 1979.

[9] Nandapurkar, P. J. and Winfree, A. T., "A Computational Study of Twisted Linked Scroll Waves in Excitable Media", Physica 29D (1987), 69-83.

[10] FitzHugh, R., "Mathematical Models of Excitation and Propagation in Nerve", Biological Engineering, H.P. Schwan (Ed), New York, McGraw-Hill, 1969.

[11] Ivanitsky, G. R., Krinsky, V. I. and Selkov, E. E., "Mathematical Biophysics of Cell", The Publishing House "Nauka", Moscow, 1978.

[12] Pertsov, A. M., Ermakova, E. A. and Panfilov, A. V., "Rotating Spiral Waves in a Modified FitzHugh-Nagumo Model", Physica 14 D (1984), 117-124.

[13] Moe, G., Rheinboldt, W. and Abildskov J., "A Computer Model of Atrial Fibrillation", American Heart Journal, 1964, v67, n2.

[14] Gul'ko, F. B. and Petrov, A. A., "On a Mathematical Model of Excitation Processes in Purkinje Fiber", Biofizika, 15(3), 1972.

```
/*****************************************************************/
/*  This file contains constants, type and macro declarations    */
/*  for the heart muscle simulation program.                     */
/*                                                               */
/*  Copyright 1988.  All rights reserved.  Alex Pang             */
/*****************************************************************/

/*****************************************************************/
/*  Constants                                                    */
/*****************************************************************/

#define FALSE           0
#define TRUE            1

#define EUL             0
#define RK2             1
#define RK4             2

#define num_eqns        2

#define xmax            128     /*  number of columns   */
#define ymax            128     /*  number of rows      */

#define A               0.12

#define col_range       80      /*  # of color per shade*/

#define mincol          80      /*  range is 0..255     */
#define maxcol          255

#define minval          -0.2    /*  range of values for */
#define maxval          1.1     /*  the E field         */

#define r_off           16
#define g_off           96
#define b_off           176

#define hole_color      0
#define dead_color      1

/*****************************************************************/
/*  Types                                                        */
/*****************************************************************/

typedef unsigned char   byte;

domain Heart {
        byte            color;          /* range from 0 .. 255  */
        float           E;              /* state variable 1     */
        float           I;              /* state variable 2     */
        float           X;              /* external current     */
        float           S;              /* sign of dE/dT        */
        float           Ival[2];        /* values for I         */
        short int       dead;           /* boolean variable     */
        short int       hole;           /* another boolean      */
        short int       stimulated;     /* yet another boolean  */
        unsigned long   row;            /* row # of processor   */
        unsigned long   col;            /* col # of processor   */
};
```

```
/*****************************************************************/
/*  This program reads in a description file for the heart       */
/*  muscle to simulate, and carries out the simulation on CM2    */
/*                                                               */
/*  Copyright 1988.  All rights reserved.  Alex Pang             */
/*****************************************************************/

#include <grid.hs>
#include <math.hs>
#include "/a/pang/Include/cmfb.h"
#include "heart.hs"

/*****************************************************************/
/*  Global variables.                                            */
/*****************************************************************/

int             method;         /* integration method           */

float           h;              /* space increment size         */
float           coefficient;    /* decoupling constant          */
float           ap_duration;    /* action potential duration    */

float           delta_t;        /* time increment size          */
float           tfinal;         /* total integration time       */

float           coef_2;         /* coefficient used in rk4      */
float           coef_6;         /* coefficient used in rk4      */

int             fde_cycle;      /* # of integrations per fde    */
int             disp_cycle;     /* # of integrations per display*/

CMFB_display_id fildes;         /* display_id                   */
CMFB_buffer_id_t bank;          /* buffer_id                    */

float           initialE;       /* initial condition value : E  */
float           initialI;       /* initial condition value : I  */

float           threshold;

float           Gr;             /* parameters for heart model   */
float           Gf;

float           E1;             /* parameters used for state    */
float           E2;             /* variable E.                  */

int             xstim;          /* x location of stimulus       */
int             ystim;          /* y location of stimulus       */
int             hstim;          /* length of 1 side of stimulus */
float           tstim;          /* duration of stimulus         */
float           stimulus;       /* strength of stimulus         */

int             ul_x, ul_y;     /* upper_left  of dead area     */
int             lr_x, lr_y;     /* lower_right of dead area     */
int             bool_dead;      /* True if any dead cells       */
int             resurrected;    /* flag for resurrect status    */
float           t_resurrect;    /* when to restore dead cells   */

int             xhole, yhole;   /* hole center coordinate       */
int             rhole;          /* radius of hole               */

float           t = 0.0;        /* current integration time step*/
```

```
GRID2_DEFINE( domain Heart, heart_grid, GRID_NEWS_ORDER, ymax, xmax )

/*****************************************************************/
/*  Forward declarations for functions                           */
/*****************************************************************/

extern int              yyparse( void );
extern void             ctable ( void );
extern void Heart::     display( void );
extern void Heart::     eul    ( void );
extern void Heart::     rk2    ( void );
extern void Heart::     rk4    ( void );
extern void Heart::     fde    ( void );

/*****************************************************************/
/*  Main program                                                 */
/*****************************************************************/

main( argc, argv )
int     argc;
char    *argv[];
{
        int             count = 0;
        void            (Heart:: *integration_step)( void );

        /*********************************************************/
        /*  Read in system description file.                     */
        /*********************************************************/

        bool_dead = FALSE;

        yyparse();

        threshold = A / Gf;

        E1 = A  / (Gr + Gf);
        E2 = (A + Gr) / (Gr + Gf);

        coef_2 = delta_t * 0.5;
        coef_6 = delta_t / 6.0;

        coefficient = coefficient / (h*h);
        stimulus    = stimulus / Gf;

        /*********************************************************/
        /*  Initialize field.                                    */
        /*********************************************************/

        [domain Heart].{
                float       distance;

                if (method == EUL)      integration_step = eul;
                if (method == RK2)      integration_step = rk2;
                if (method == RK4)      integration_step = rk4;

                color       = 0;
                E           = initialE;
                I           = initialI;
                X           = 0.0;
```

```
        Ival[0]     = ap_duration;
        Ival[1]     = 0.5;
        dead        = FALSE;
        hole        = FALSE;
        stimulated  = FALSE;
        row         = GRID2_MY_INDEX_0( heart_grid);
        col         = GRID2_MY_INDEX_1( heart_grid);

        /*************************************************/
        /*  Set dead cells to true.                      */
        /*************************************************/

        if ( bool_dead * ((row >= ul_y) && (row <= lr_y) &&
                          (col >= ul_x) && (col <= lr_x)) ) {
                dead = TRUE;
                E    = 0.0; }

        /*************************************************/
        /*  Set stimulated cells to true.                */
        /*************************************************/

        if ((col >= xstim) && (col <= xstim + hstim) &&
            (row >= ystim) && (row <= ystim + hstim))
                stimulated = TRUE;

        /*************************************************/
        /*  Set hole-cells to true.                      */
        /*************************************************/

        distance = (col-xhole)*(col-xhole) + (row-yhole)*(row-yhole);
        if ( sqrt(distance) <= rhole ) {
                hole = TRUE;
                E    = 0.0; }

        /*************************************************/
        /*  Set boundary cells to be hole-cells.         */
        /*************************************************/

        if ((row == 0) || (row == ymax-1) ||
            (col == 0) || (col == xmax-1)) {
                hole = TRUE;
                E    = 0.0; }
};

/*******************************************************/
/*  Setup CM framebuffer and color table.              */
/*******************************************************/

ctable();
```

```
/*******************************************************/
/*  Do it, baby.                                       */
/*******************************************************/

resurrected = FALSE;

[domain Heart].{
        while( t <= tfinal ) {

                if (count % disp_cycle == 0)    display();

                (*integration_step)();

                if (count % fde_cycle == 0)     fde();

                t += delta_t;
                count++;

                /*  resurrect if necessary      */
                if ((!resurrected) && (t >= t_resurrect)) {
                        dead = FALSE;
                        resurrected = TRUE; }
        }
}
}
```

```
/*****************************************************************/
/*  This procedure sets up the color table for action potentials*/
/*                                                               */
/*  Copyright 1988.  All rights reserved.  Alex Pang             */
/*****************************************************************/

#include "/a/pang/Include/cmfb.h"
#include "heart.hs"

extern CMFB_display_id  fildes;
extern CMFB_buffer_id_t bank;

void    ctable()
{
        int             i;
        float           factor;
        char            *names[16];
        byte            colortable[maxcol+1];

        CMFB_available_displays( names );
        CMFB_attach_display( names[0], &fildes );
        CMFB_initialize_display( &fildes, 8, 1 );

        factor = (float) (maxcol - mincol) / col_range;

        /*******************************************************/
        /*  Red Bank: entries 16-95                            */
        /*******************************************************/

        for( i=0; i<=maxcol; i++ ) colortable[i] = 0;

        colortable[dead_color] = maxcol;                /* pink */

        for( i=r_off; i<col_range+r_off; i++ )          /* red  */
                colortable[i] = (i-r_off) * factor + mincol;

        CMFB_write_color_table( &fildes, CMFB_red,   colortable );

        /*******************************************************/
        /*  Green Bank: entries 96-175                         */
        /*******************************************************/

        for( i=0; i<=maxcol; i++ ) colortable[i] = 0;

        colortable[dead_color] = 0;                     /* pink */

        for( i=g_off; i<col_range+g_off; i++ )          /* green*/
                colortable[i] = (i-g_off) * factor + mincol;

        CMFB_write_color_table( &fildes, CMFB_green, colortable );

        /*******************************************************/
        /*  Blue Bank: entries 176-255                         */
        /*******************************************************/

        for( i=0; i<=maxcol; i++ ) colortable[i] = 0;

        colortable[dead_color] = maxcol;                /* pink */

        for( i=b_off; i<col_range+b_off; i++ )          /* blue */
                colortable[i] = (i-b_off) * factor + mincol;

        CMFB_write_color_table( &fildes, CMFB_blue,  colortable );

        /*******************************************************/
        /* Set up bank, pan and zoom, and overlay              */
        /*******************************************************/

        bank  = CMFB_current_buffer( &fildes );

        CMFB_set_pan ( &fildes, -60, -32 );     /*  WATCH IT!!  */
        CMFB_set_zoom( &fildes,  4, 4, 0 );
}
```

```
/*****************************************************************/
/*  This function displays the contents of the field array.      */
/*                                                               */
/*  Copyright 1988.  All rights reserved.  Alex Pang             */
/*****************************************************************/

#include <grid.hs>
#include "heart.hs"
#include "/a/pang/Include/cmfb.h"

/*****************************************************************/
/*  Externs                                                      */
/*****************************************************************/

extern float            threshold;

extern CMFB_display_id  fildes;
extern CMFB_buffer_id_t bank;

GRID2_EXTERN( domain Heart, heart_grid, GRID_NEWS_ORDER, ymax, xmax )

/*****************************************************************/
/*  Display program                                              */
/*****************************************************************/

void    Heart::display()
{
        if      ( hole )
                color = hole_color;
        else if ( dead )
                color = dead_color;
        else if ( E < threshold )
                color = col_range*(E-minval)/(threshold-minval) + b_off;
        else if ( S <= 0 )
                color = col_range*(E-threshold)/(maxval-threshold) + g_off;
        else    color = col_range*(E-threshold)/(maxval-threshold) + r_off;

        CMFB_write_always( &fildes, bank,  &color,  0, 0 );
}
```

```
/*****************************************************************/
/*  This file contains integration routines.                     */
/*                                                               */
/*  Copyright 1989.  All rights reserved.  Alex Pang             */
/*****************************************************************/

#include <grid.hs>
#include "heart.hs"

/*****************************************************************/
/*  Externs                                                      */
/*****************************************************************/

extern  float           delta_t;
extern  float           coef_2;
extern  float           coef_6;

extern  float           Heart::fE( float, float );
extern  float           Heart::fI( float, float );

GRID2_EXTERN( domain Heart, heart_grid, GRID_NEWS_ORDER, ymax, xmax )

/*****************************************************************/
/*  Start of explicit euler                                      */
/*****************************************************************/

void    Heart::eul()
{
        float           newE;                   /*  tmp E variable      */

        newE = E + delta_t * fE( E, I );
        I    = I + delta_t * fI( E, I );
        S    = newE - E;
        E    = newE;
}

/*****************************************************************/
/*  Start of runge-kutta-2                                       */
/*****************************************************************/

void    Heart::rk2()
{
        float           k00;                    /*  1st subscript is    */
        float           k01;                    /*  for state equation. */

        float           k10;                    /*  2nd subscript is    */
        float           k11;                    /*  for rk2.            */

        float           newE;                   /*  tmp E variable      */
        float           dE;
        float           dI;

        /*  Computes the k0's of the 2 state equations. */

        k00 = fE( E, I );
        k10 = fI( E, I );

        /*  Computes the k1's of the 2 state equations. */

        dE = E + delta_t * k00;
        dI = I + delta_t * k10;

        k01 = fE( dE, dI );
        k11 = fI( dE, dI );

        newE = E + coef_2 * ( k00 + k01 );
        I    = I + coef_2 * ( k10 + k11 );
        S    = newE - E;
        E    = newE;
}

/*****************************************************************/
/*  Start of runge-kutta-4                                       */
/*****************************************************************/

void    Heart::rk4()
{
        float           k00;                    /*  1st subscript is    */
        float           k01;                    /*  for state equation. */
        float           k02;                    /*  2nd subscript is    */
        float           k03;                    /*  for rk4.            */

        float           k10;
        float           k11;
        float           k12;
        float           k13;

        float           newE;                   /*  tmp E variable      */
        float           dE;
        float           dI;

        /*  Computes the k0's of the 2 state equations. */

        k00 = fE( E, I );
        k10 = fI( E, I );

        /*  Computes the k1's of the 2 state equations. */

        dE = E + coef_2 * k00;
        dI = I + coef_2 * k10;

        k01 = fE( dE, dI );
        k11 = fI( dE, dI );

        /*  Computes the k2's of the 2 state equations. */

        dE = E + coef_2 * k01;
        dI = I + coef_2 * k11;

        k02 = fE( dE, dI );
        k12 = fI( dE, dI );

        /*  Computes the k3's of the 2 state equations. */

        dE = E + delta_t * k02;
        dI = I + delta_t * k12;

        k03 = fE( dE, dI );
        k13 = fI( dE, dI );

        newE = E + coef_6 * ( k00 + 2*(k01 + k02) + k03 );
        I    = I + coef_6 * ( k10 + 2*(k11 + k12) + k13 );
        S    = newE - E;
        E    = newE;
}
```

```
/*****************************************************************/
/*  This file computes the two state equation for this model.    */
/*                                                               */
/*  Copyright 1989.  All rights reserved.  Alex Pang             */
/*****************************************************************/

#include <grid.hs>
#include "heart.hs"

/*****************************************************************/
/*  Externs                                                      */
/*****************************************************************/

extern  float           t;

extern  float           Gr;
extern  float           Gf;

extern  float           E1;
extern  float           E2;

extern  float           tstim;
extern  float           stimulus;

GRID2_EXTERN( domain Heart, heart_grid, GRID_NEWS_ORDER, ymax, xmax )

/*****************************************************************/
/*  Evaluates the function associated to the state variable E.   */
/*****************************************************************/

float   Heart::fE( ee, ii )
float   ee, ii;
{
        float   tmp;
        float   stim;

        if (ee < E1)
                tmp = Gr * ee;
        else    if (ee <= E2 )
                        tmp = -Gf * ee + A;
                else    tmp =  Gr * (ee - 1);

        stim = ((t <= tstim) && (stimulated)) * stimulus;

        return( (X - tmp - ii + stim) * !dead * !hole );
}

/*****************************************************************/
/*  Evaluates the function associated to the state variable I.   */
/*****************************************************************/

float   Heart::fI( ee, ii )
float   ee, ii;
{
        return( (ee - ii) * Ival[(ee <= 0.01)] );
}
```

```
/*****************************************************************/
/*  This function computes the finite difference equation at     */
/*  the end of every "fde.cycle" integration steps.              */
/*                                                               */
/*  Copyright 1988.  All rights reserved.  Alex Pang             */
/*****************************************************************/

#include <grid.hs>
#include "heart.hs"

/*****************************************************************/
/*  Externs                                                      */
/*****************************************************************/

extern  float   coefficient;

GRID2_EXTERN( domain Heart, heart_grid, GRID_NEWS_ORDER, ymax, xmax )

/*****************************************************************/
/*  Start of finite difference equation                          */
/*****************************************************************/

void    Heart::fde()
{
        float   El, Er, Eu, Ed;
        float   Enews[5];

        /*  First obtain values of E and its neighbors  */

        Enews[0] = E;
        Enews[1] = GRID2_GET_FROM_WEST (heart_grid, &E, GRID_DONT_CARE);
        Enews[2] = GRID2_GET_FROM_EAST (heart_grid, &E, GRID_DONT_CARE);
        Enews[3] = GRID2_GET_FROM_NORTH(heart_grid, &E, GRID_DONT_CARE);
        Enews[4] = GRID2_GET_FROM_SOUTH(heart_grid, &E, GRID_DONT_CARE);

        /*  If cell is beside a hole (ie at boundary), flux is 0  */

        El = Enews[ !GRID2_GET_FROM_WEST (heart_grid,&hole,GRID_DONT_CARE)      ];
        Er = Enews[ !GRID2_GET_FROM_EAST (heart_grid,&hole,GRID_DONT_CARE) * 2 ];
        Eu = Enews[ !GRID2_GET_FROM_NORTH(heart_grid,&hole,GRID_DONT_CARE) * 3 ];
        Ed = Enews[ !GRID2_GET_FROM_SOUTH(heart_grid,&hole,GRID_DONT_CARE) * 4 ];

        /*   If beside dead cell, return Ex = 0  */

        El *= !GRID2_GET_FROM_WEST (heart_grid, &dead, GRID_ZERO);
        Er *= !GRID2_GET_FROM_EAST (heart_grid, &dead, GRID_ZERO);
        Eu *= !GRID2_GET_FROM_NORTH(heart_grid, &dead, GRID_ZERO);
        Ed *= !GRID2_GET_FROM_SOUTH(heart_grid, &dead, GRID_ZERO);

        X = coefficient * !dead * !hole * ( El + Er + Eu + Ed - 4*E );
}
```

```
%{
/* lex definitions */
%}

#       [0-9]

%%
%{
/* c definitions needed by: lex.yy.c    */
%}

        int     atoi();
        float   atof();

%{
/* lex rules */
%}

integration                     return( T_METHOD        );
EUL                             return( T_METHOD_EUL    );
RK2                             return( T_METHOD_RK2    );
RK4                             return( T_METHOD_RK4    );

dimension"."h                   return( T_DIM_H         );
membrane"."coef                 return( T_MEM_COEF      );
ap"."duration                   return( T_AP_DURATION   );

time"."delta                    return( T_TIME_DELTA    );
time"."final                    return( T_TIME_FINAL    );

fde"."cycle                     return( T_FDE_CYCLE     );
display"."cycle                 return( T_DISP_CYCLE    );

initial"."E                     return( T_INITIAL_E     );
initial"."I                     return( T_INITIAL_I     );

Gr                              return( T_GR            );
Gf                              return( T_GF            );

stimulus"."x                    return( T_STIMULUS_X    );
stimulus"."y                    return( T_STIMULUS_Y    );
stimulus"."h                    return( T_STIMULUS_H    );
stimulus"."t                    return( T_STIMULUS_T    );
stimulus                        return( T_STIMULUS      );

upper_left"."dead               return( T_UL_DEAD       );
lower_right"."dead              return( T_LR_DEAD       );

time"."resurrect                return( T_TIME_RESURRECT);

hole"."center                   return( T_H_CENTER      );
hole"."radius                   return( T_H_RADIUS      );

(\+|\-)?[#]+                    {
                                yylval.ival = atoi( yytext );
                                return( T_INTEGER );
                                }

(\+|\-)?[#]*"."[#]*             {
                                yylval.fval = atof( yytext );
                                return( T_FLOAT );
                                }

(\+|\-)?[#]*"."[#]+             {
                                yylval.fval = atof( yytext );
                                return( T_FLOAT );
                                }

[ \n\t:,]                       ;       /* ignore these         */
"/**[^**/*]***/*"               ;       /* ignore comments      */

%%

/* user routines */

yywrap()
{
        return(1);
}
```

```
%{
/* This section obtained from global section of main.c  */

#include <stdio.h>

#define EUL     0               /* these are from heart.hs      */
#define RK2     1
#define RK4     2

int             method;         /* integration method           */

float           h;              /* space increment size         */
float           coefficient;    /* decoupling constant          */
float           ap_duration;    /* action potential duration    */

float           delta_t;        /* time increment size          */
float           tfinal;         /* totalintegration time        */

int             fde_cycle;      /* # of integrations per fde    */
int             disp_cycle;     /* # of integrations per display*/

float           initialE;       /* initial condition value : E  */
float           initialI;       /* initial condition value : I  */

float           Gr;             /* parameters for heart model   */
float           Gf;

int             xstim;          /* x location of stimulus       */
int             ystim;          /* y location of stimulus       */
int             hstim;          /* length of 1 side of stimulus */
float           tstim;          /* duration of stimulus         */
float           stimulus;       /* strength of stimulus         */

int             bool_dead;      /* True if any dead cells       */
int             ul_x, ul_y;     /* upper_left  of dead area     */
int             lr_x, lr_y;     /* lower_right of dead area     */
float           t_resurrect;    /* when to restore dead cells   */

int             xhole, yhole;   /* hole center coordinate       */
int             rhole;          /* radius of hole               */

%}

/* yacc declarations */

%union  {
        int     ival;
        float   fval;
        }

%token  T_METHOD
%token  T_METHOD_EUL
%token  T_METHOD_RK2
%token  T_METHOD_RK4

%token  T_DIM_H
%token  T_MEM_COEF
%token  T_AP_DURATION

%token  T_TIME_DELTA
%token  T_TIME_FINAL

%token  T_FDE_CYCLE
%token  T_DISP_CYCLE

%token  T_INITIAL_E
%token  T_INITIAL_I

%token  T_GR
%token  T_GF

%token  T_STIMULUS_X
%token  T_STIMULUS_Y
%token  T_STIMULUS_H
%token  T_STIMULUS_T
%token  T_STIMULUS

%token  T_UL_DEAD
%token  T_LR_DEAD
%token  T_TIME_RESURRECT

%token  T_H_CENTER
%token  T_H_RADIUS

%token  <fval>  T_FLOAT
%token  <ival>  T_INTEGER

%start  Start

%%

/* yacc grammar rules */

Start           : Integration Conditions
                ;

Integration     : Method DimensionH Coefficient APDuration TimeDelta TimeFinal Interval
                ;

Method          : T_METHOD      T_METHOD_EUL
                        {       method = EUL;
                        }
                | T_METHOD      T_METHOD_RK2
                        {       method = RK2;
                        }
                | T_METHOD      T_METHOD_RK4
                        {       method = RK4;
                        }
                ;

DimensionH      : T_DIM_H       T_FLOAT
                        {       h = $2;
                        }
                ;

Coefficient     : T_MEM_COEF    T_FLOAT
                        {       coefficient = $2;
                        }
                ;
```

```
APDuration      : T_AP_DURATION T_FLOAT
                        {       ap_duration = $2;
                        }
                ;

TimeDelta       : T_TIME_DELTA  T_FLOAT
                        {       delta_t = $2;
                        }
                ;

TimeFinal       : T_TIME_FINAL  T_FLOAT
                        {       tfinal = $2;
                        }
                ;

Interval        : T_FDE_CYCLE   T_INTEGER
                  T_DISP_CYCLE  T_INTEGER
                        {       fde_cycle  = $2;
                                disp_cycle = $4;
                        }
                ;

Conditions      : Initial Parameters Stimulus
                | Initial Parameters Stimulus DeadArea
                | Initial Parameters Stimulus DeadArea Duration
                | Initial Parameters Stimulus Hole
                | Initial Parameters Stimulus DeadArea Hole
                | Initial Parameters Stimulus DeadArea Duration Hole
                ;

Initial         : T_INITIAL_E   T_FLOAT
                  T_INITIAL_I   T_FLOAT
                        {       initialE = $2;
                                initialI = $4;
                        }
                ;

Parameters      : T_GR          T_FLOAT
                  T_GF          T_FLOAT
                        {       Gr = $2;
                                Gf = $4;
                        }
                ;

Stimulus        : T_STIMULUS_X  T_INTEGER
                  T_STIMULUS_Y  T_INTEGER
                  T_STIMULUS_H  T_INTEGER
                  T_STIMULUS_T  T_FLOAT
                  T_STIMULUS    T_FLOAT
                        {       xstim    = $2;
                                ystim    = $4;
                                hstim    = $6 - 1;
                                tstim    = $8;
                                stimulus = $10;
                        }
                ;

DeadArea        : T_UL_DEAD     T_INTEGER       T_INTEGER
                  T_LR_DEAD     T_INTEGER       T_INTEGER
                        {       ul_x        = $2;
                                ul_y        = $3;
                                lr_x        = $5;
                                lr_y        = $6;
                                bool_dead   = 1;
                                t_resurrect = tfinal;
                        }
                ;

Duration        : T_TIME_RESURRECT      T_FLOAT
                        {       t_resurrect = $2;
                        }
                ;

Hole            : T_H_CENTER    T_INTEGER       T_INTEGER
                  T_H_RADIUS    T_INTEGER
                        {       xhole = $2;
                                yhole = $3;
                                rhole = $5;
                        }
                ;

%%

/* user routines */
#include "lex.yy.c"

yyerror( s )
char    *s;
{
        fprintf( stderr, "bummer: %s on line %d of system description file.\n",
                s, yylineno );
        exit(-1);
}
```

1.normal


```
integration     :       RK2     /* integration method           */

dimension.h     :       0.6     /* spacing between grid points  */
membrane.coef   :       0.5     /* speed of propagation         */
ap.duration     :       0.03    /* width of potential (green)   */

time.delta      :       0.03    /* integration time step        */
time.final      :       250.0   /* duration of simulation       */

fde.cycle       :       2       /* # of integrations per fde    */
display.cycle   :       20      /* # of integrations per display*/

initial.E       :       0.0
initial.I       :       0.0

Gr              :       30.0
Gf              :       0.735

stimulus.x      :       1       /* x coord of upperleft of stim */
stimulus.y      :       1       /* y coord of upperleft of stim */
stimulus.h      :       4       /* size of stimulus is # by #   */
stimulus.t      :       4.0     /* duration of stimulus         */
stimulus        :       0.3     /* must be greater than 0.12    */
```

2.spiral


```
integration      :      RK2      /* integration method           */

dimension.h      :      0.6      /* spacing between grid points  */
membrane.coef    :      0.5      /* speed of propagation         */
ap.duration      :      0.03     /* width of potential (green)   */

time.delta       :      0.03     /* integration time step        */
time.final       :      500.0    /* duration of simulation       */

fde.cycle        :      2        /* # of integrations per fde    */
display.cycle    :      20       /* # of integrations per display*/

initial.E        :      0.0
initial.I        :      0.0

Gr               :      30.0
Gf               :      0.735

stimulus.x       :      1        /* x coord of upperleft of stim */
stimulus.y       :      1        /* y coord of upperleft of stim */
stimulus.h       :      4        /* size of stimulus is # by #   */
stimulus.t       :      4.0      /* duration of stimulus         */
stimulus         :      0.3      /* must be greater than 0.12    */

upper_left.dead  :      55       0
lower_right.dead:       60       60

time.resurrect   :      110.0    /* restore dead cells after this time   */
```

```
integration      :      RK2     /* integration method           */

dimension.h      :      0.6     /* spacing between grid points  */
membrane.coef    :      0.5     /* speed of propagation         */
ap.duration      :      0.03    /* width of potential (green)   */

time.delta       :      0.03    /* integration time step        */
time.final       :      500.0   /* duration of simulation       */

fde.cycle        :      2       /* # of integrations per fde    */
display.cycle    :      20      /* # of integrations per display*/

initial.E        :      0.0
initial.I        :      0.0

Gr               :      30.0
Gf               :      0.735

stimulus.x       :      1       /* x coord of upperleft of stim */
stimulus.y       :      1       /* y coord of upperleft of stim */
stimulus.h       :      4       /* size of stimulus is # by #   */
stimulus.t       :      4.0     /* duration of stimulus         */
stimulus         :      0.3     /* must be greater than 0.12    */

upper_left.dead  :      50      40
lower_right.dead:       54      70

time.resurrect   :      100.0   /* restore dead cells after this time   */
```

4.hole


```
integration      :      RK2      /* integration method           */

dimension.h      :      0.6      /* spacing between grid points  */
membrane.coef    :      0.5      /* speed of propagation         */
ap.duration      :      0.03     /* width of potential (green)   */

time.delta       :      0.03     /* integration time step        */
time.final       :      500.0    /* duration of simulation       */

fde.cycle        :      2        /* # of integrations per fde    */
display.cycle    :      20       /* # of integrations per display*/

initial.E        :      0.0
initial.I        :      0.0

Gr               :      30.0
Gf               :      0.735

stimulus.x       :      45       /* x coord of upperleft of stim */
stimulus.y       :      45       /* y coord of upperleft of stim */
stimulus.h       :      4        /* size of stimulus is # by #   */
stimulus.t       :      4.0      /* duration of stimulus         */
stimulus         :      0.3      /* must be greater than 0.12    */

upper_left.dead  :      0        60
lower_right.dead:       64       68

time.resurrect   :      75.0     /* restore dead cells after this time   */

hole.center      :      64       64
hole.radius      :      10
```