I. Philosophy of EZ-Spiral

EZ-Spiral is a simplified version of the C program that I use to study the dynamics of spiral waves in excitable media [1-4,6]. EZ-Spiral is designed to be as simple as possible while offering what I believe to be the most important features for spiral simulations. These include:

(1) Continuous range of spatio-temporal resolutions.

(2) Choice of implicit or explicit time stepping of the reaction kinetics.

(3) Choice of ratio of diffusion coefficients.

(4) Computation of spiral-tip location.

(5) Automatic generation of initial conditions.

(6) Interactive graphics with control of spiral location.

The code should be of use to those who are interested in the serious study of excitable media as well as those who simply want to generate pretty pictures. Most of the documentation is contained in the code itself. The notation for the model reaction-diffusion equations and a description of parameters are given in Sec. IV.

Changes in Version 2. Version 2 of EZ-Spiral fixes many of the awkward properties of Version 1. In particular, almost all parameters one routinely wants to change are now contained in *task.dat* and are read in at run time. This eliminates most of the recompiling that was necessary with Version 1. The other major change is that the graphics subroutines have been rewritten using X11 calls and hence the code is portable to a large number of machines. Elementary graphics are even available for black and white X-terminals. Finally, parts of the code have been cleaned up and simplified. By editing just two lines it is now possible to simulate a large variety of models with EZ-Spiral (see Sec. IV). I include at the end (Sec. V) a simple recipe for obtaining turbulent spirals. Note that the format of *task.dat* has changed slightly from Version 1. The format of the field files has not changed.

Changes in Version 2.3 The primary change in Version 2.3 is that the the higher-order implicit u time stepping is implemented. See [6].

Changes in Version 3 Version 3 comes from a large re-write of the code. It in fact was obtained from reducing EZ-Scroll to the 2D case. I decided to keep separate 2D and 3D version of the code. The main new features are that the 2D domain can now be a rectangle and the graphics is heavily OpenGL based, with primitive X11 calls only used for opening the window and for event handling. The format of *task.dat* has changed from Version 2.

II. Running EZ-Spiral

Files: You should have the following files: *ezspiral.c, ezstep.c, eztip.c, ezgraphGL.c, ezspiral.h, ezstep.h ezgraphGL.h, task.dat*, and *Makefile*. You will probably want to save copies of these files (in compressed tar format).

make: It is up to you to edit *Makefile* as necessary for your system. You need an ANSI C compiler and for graphics you need X11 and OpenGL. If you run on an up-to-date Linux system you should have everything you need. Typing *make* will make *ezspiral*. If you edit one of the source or header files you must remake EZ-Spiral.

Running: Make *ezspiral* by typing *make*. Then run by typing *ezspiral*. A window should open. With the pointer in the window type a *space* on the keyboard. A spiral should be generated. The v-field is plotted. This is a coarse resolution run showing the speed possible with EZ-Spiral simulations. With the pointer in the EZ-Spiral window, you can move the spiral around using the arrow keys on the keyboard. Center the spiral in the box.

A simulation will finish when either:

the number of time steps set in *task.dat* is taken,
or you type q or Q with the pointer in the spiral window.
or you type *Esc* with the pointer in the spiral window
or you use the menu produced by the window manager (assuming there is one) to kill the graphics window.

The last two possibilities will result in a hard exit with nothing saved.

After a successful run with a soft exit (the first two cases above), you will have a file *fc.dat* in your directory which contains the final conditions of the run (either at the final time or at the type you stopped the simulation with q or Q). If you copy this file to *ic.dat*, then the next time you run *ezspiral*, this file will be read and used as an initial condition.

To get some insight into what the code can do, try the following runs in order:

1. In *task.dat*, change nx and ny from 101 to 201, change Time steps per plot to 16, and change verbose from 2 to 1 (the normal value). Copy *fc.dat* from the run with nx=nx=101 to *ic.dat*. Run *ezspiral*. This is a more accurate run and is getting close to a fully resolved simulation of the underlying PDEs.

2. While the code is running put the pointer in the spiral window. Typing either p or P within the spiral window will pause the simulation until you type a *space* on the keyboard. Typing t will toggle tip plotting. Typing u will show the u-field rather than the v-field. You can go back to the v-field by typing v. Typing n will show no field. You can, however, still plot the tip path if you desire. *Note that for coarse simulations, tip finding can produce uncertain results and will probably fail.*

3. In *task.dat* change write_tip from 0 to 1 and time steps per write to history file to 10. Now run run *ezspiral* again. After this run you will have a file *tip.dat* with the path of the spiral tip (t, x_{tip}, y_{tip}), and a file *history.dat* with the time series of values (t, u, v) at the grid point $(10, 20)$. You can set the history point to some other value in *task.dat*.

4. Change the kinetics parameter a in *task.dat* from 0.75 to 0.62. Run again. With these parameters the spiral will "meander". You can see this best by plotting the tip.

You should have the hang of it by now.

III. Further Comments on EZ-Spiral

It is expected that you will modify the code as needed for your purposes. The code should be more or less self-explanatory. The bulk of the code is devoted to I/O, graphics, and tip finding. Almost all of the execution time is spent on just a few lines of code in the routine step() in *ezstep.c*.

Note that I have used many external (i.e. global) variables and have not used any structures. I thought that this would make the code more readable, and therefore usable, to those not entirely familiar with C. I believe that the crucial for loops in routine step() compile to very efficient code. See comments in *ezstep.h*.

I make no claims as to what constitutes a fully resolved simulation of the underlying PDEs. If you are concerned about this I would recommend having at least 6 points across the interface (black in the u-field plot) and a ratio of $dt/\epsilon < 0.5$. You might also decrease delta by an

order of magnitude or more. You should experiment for yourself. Note that for $dt/\epsilon < 1$ you can use either the explicit or the implicit form of the reaction kinetics. The explicit form is slightly faster, but the new implicit form [6] is definitely more accurate and I highly recommend it. For delta very small, you should remove the if from the for loops in step().

In general I believe in the 9-point formula for the Laplacians [1,6], rather than the 5-point formula. The 9-point formula maintains rotational symmetry to leading order. For coarse-grained simulations with $D_v = 0$ the difference in execution time is small compared with what is gained. If you are unconvinced, run the simulation described in (1) with both 5-point and 9-point Laplacians.

I believe tip finding is quite reliable for well resolved simulations. For coarse-grained simulations it may fail miserably.

With GRAPHICS in *ezspiral.h* set to 0, none of the X11 graphics is compiled and this should allow the code to run on *any* machine. However, do not run EZ-Spiral on a vector machine without appropriate modification.

IV. Equations and Parameters

The model reaction-diffusion equations are [1,2]:

$$\frac{\partial u}{\partial t} = \nabla^2 u + \epsilon^{-1} u(1-u)(u - u_{th}(v)), \quad \frac{\partial v}{\partial t} = D_v \nabla^2 v + g(u,v)$$

where in the simplest case

$$u_{th}(v) = \frac{v+b}{a}, \quad g(u,v) = u - v,$$

so that a, b, and ϵ are parameters of the reaction kinetics. D_v is the ratio of diffusion coefficients ($D_u \equiv 1$ by choice of length scales). In addition, L_x and L_y are the lengths of the sides. Thus the "physical" parameters in the simulation are: a, b, ϵ, L_x, L_y, and D_v.

The method employed in EZ-Spiral is (essentially) independent of the choice of the functions $u_{th}(v)$ and $g(u,v)$ and the user is free to set these to whatever is desired. See *ezstep.h*.

The "numerical" parameters for the simulation are: nx, ny, ts (= time step as fraction of the diffusion stability limit), and delta (= small numerical parameter [1,2]). The other parameters set in *task.dat* are more or less self-explanatory.

Depending on the hardware you have, plotting can take a significant amount of time, so choose Time steps per plot judiciously. Also note that plotting the u-field is faster than plotting the v-field. See the code.

V. Spiral Turbulence with EZ-Spiral

I include in *ezstep.h* two choices for the slow kinetics $g(u, v)$ other than the choice $g(u, v) = u - v$. These are $g(u, v) = u^3 - v$ and the function given by Bär and Eiswirth[5]. Both choices produce spiral breakup followed by spiral "turbulence". Here I give a simple recipe for spiral turbulence.

In *ezstep.h* turn off the "standard model" by putting #if 0 and turn on the "simple model for turbulent spirals" by putting #if 1. Remake ezspiral. Then in *task.dat* set: a=0.75, b=0.06, 1/eps=12.0, L=80, Dv=0, nx=ny=121, ts = 0.8, and delta=1.e-4. Set the graphics parameters to your choice. Now run ezspiral. You should see spiral breakup followed by spiral turbulence. This simulation is only qualitatively correct. For a more quantitative and convincing example, set 1/eps=14 and ndim=241 (and all other parameters as before). This time you will see the formation of a single spiral. After running sufficiently long that you are convinced that the spiral is stable, stop the simulation and move *fc.dat* to *ic.dat*. Set 1/eps=12 and rerun. You will now see the spiral breakup. You can increase the spatial resolution still further if you desire and you can take several small steps in 1/eps if you wish, but I believe you will get essentially the same result.

References

[1] D. Barkley, M. Kness, and L.S. Tuckerman, Phys. Rev. A **42**, '2489 (1990).

[2] D. Barkley, Physica **49D**, 61 (1991).

[3] D. Barkley, Phys. Rev. Lett **68**, 2090 (1992).

[4] D. Barkley, Phys. Rev. Lett. **72**, 164 (1994).

[5] M. Bär and M. Eiswirth, Phys. Rev. E **48**, 1635 (1993).

[6] M. Dowel, R.M. Mantel and D. Barkley, Int. J. Bif. Chaos, 7(11) 2529–2546 (1997).

Please send comments to D.Barkley@warwick.ac.uk