REPORT ON "THE DYNAMIC STRUCTURE UNDERLYING SUBTHRESHOLD OSCILLATORY ACTIVITY AND THE ONSET OF SPIKES IN A MODEL OF MEDIAL ENTORHINAL CORTEX STELLATE CELLS"

TIMOTHY JONES

Abstract. The entorhinal cortex (EC) has been found to be a major information hub for the hippocampus [1] and seems to play an important role in memory. This part of the brain has been found to be among the first and most severely damaged by Alzheimer's disease as well as other dementing disorders; young individuals with the Alzheimer's related variant of the ApoE gene are statistically found to have thinner entorhinal cortexes than do controls [2, 3].

A common neuron found in the EC is the spiny stellate cells. These cells demonstrate, individually, subthreshold oscillation (STO) [4]. Rotstein, Opperman, White, and Kopell [5] discuss a particular conductance based model for these cells [6] that reproduces the STO phenomenon. They seek to explain the STOs in dynamical terms. The goal of this report is to summarize and simulate their results as a final project for Math 723.

1. Subthreshold Oscillations Discovered and Modeled

1.1. **Experimental results.** The thrust of this report is dynamical, not physiological, but we briefly discuss some physiological results from stellate cells of layer II in the entorhinal cortex (ECIIsc). Alonso and Llinás demonstrated these STOs in their 1989 Nature article [4] from which we present in Figure 1.1.

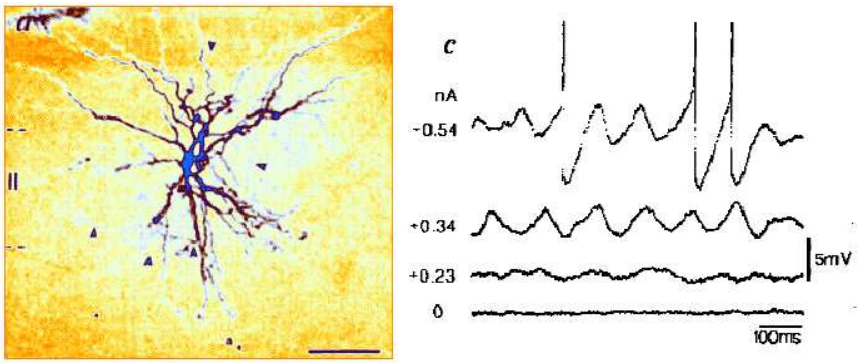


Figure 1.1. ECIIsc cell (left) under microscopic magnification. The scale bar in the lower right of the left panel is 100μm. The graph on the right represents experimental data from the neuron stimulated with current. The oscillation becomes apparent at $\tilde{5}$7mV, which the model we discuss replicates quite well. These figures are from [4]. False color introduced by this student to enhance figure.

1.2. **Conductance model.** Acker et al. [6] introduced a conductance model for the ECIIsc cells. Before going into detail about this model, we demonstrate its success in modeling the ECIIsc cells by including some graphs (Figure 1.2) of its simulation, programed by this student in MATLAB. We conclude with 3D images of the trajectory for $I_{app} = -2.5$ as modeled in the ray tracer program POV-Ray.

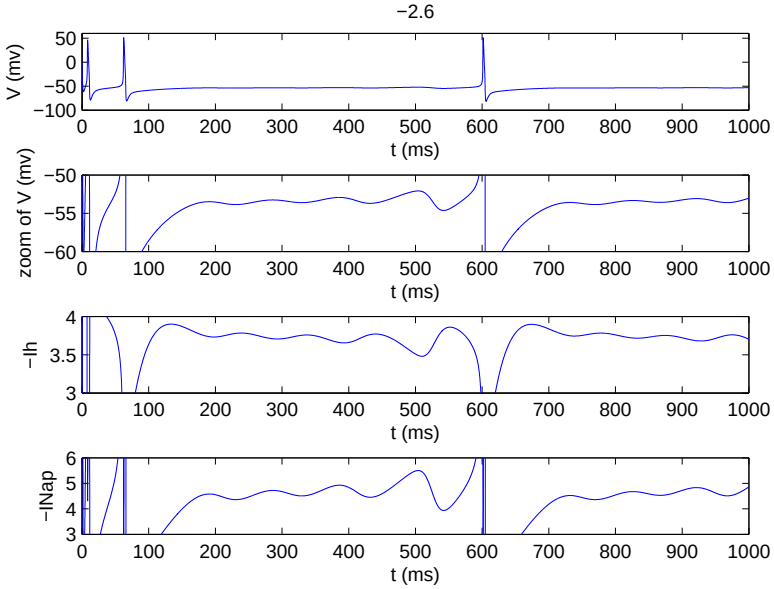


FIGURE 1.2. ECIIsc cell conductance model of Acker et al. (2003) [6].From top to bottom: Action potentials; second panel zooms in on the first one where subthreshold oscillation is apparent. The third and forth panels show the behavior of I_h and I_{Nap} whose role in the model is explored in this report. These figures were made via this student's MATLAB simulation Acker_Fig002.m (included at the end of this report) based on this model.

Channels without inactivation gates are called **persistent**, channels with such gates are called **transient**.

In a generic model, the current passing through a membrane due to a particular type of current is given as $I = \bar{g}p(V - E)$ where p is the average proportion of channels in the open state (a probability, basically), $\bar{g}$ is the maximal conductance, and E is the reverse potential of the current. In Hodgkin-Huxley type models, m (and n for K^+ and CL^- channels) represents the probability that an activation gate is in its open state, and h the probability that an inactivation gate is in the open state. See Figure 1.3 for a simple schematic of these states.

The dynamics of the activation variables such as m are covered in [7] and references therein. They are created to model experimental results. The dynamical system discussed in this report has similarly been tailored to match experimental results. The Acker et al. model is a seven dimensional system based on the interactions of the following current channels: persistent sodium (I_{Nap}), a two-component (fast and slow) hyperpolerization-activated current (I_h), and the standard Hodgkin-Huxley currents, i.e., sodium (I_{Na}), potassium (I_K) and the leak current (I_L). I_{app} is the applied bias current (DC, $\mu A/cm^2$). C is the membrane capacitance $/cm^2$. We denote the membrane potential V (mV). The remaining currents are modeled with the following equations:

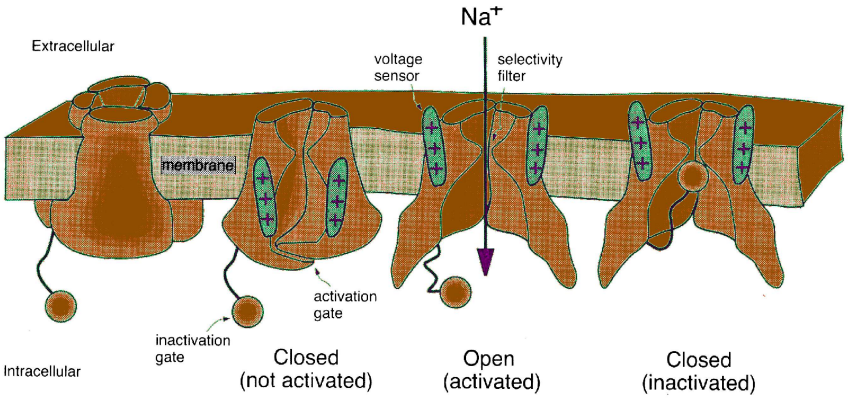


FIGURE 1.3. Figure modified from Izhikevich [7] from Armstrong and Hille 1998.

$$
\begin{aligned}
I_{Na} &= G_{Na} m^3 h (V - E_{Na}) \\
I_k &= G_K n^4 (V - E_k) \\
I_L &= G_L (V - E_L) \\
I_{Nap} &= G_p p (V - E_{Na}) \\
I_h &= G_h (0.65 r_f + 0.35 r_s)(V - E_h)
\end{aligned}
$$

The dynamical equations are:

$$C\frac{dV}{dt} = I_{app} - I_{N_a} - I_k - I_L - I_h - I_{Nap} \tag{1.1}$$

$$\frac{dm}{dt} = \frac{m_\infty(V) - m}{\tau_m(V)} \tag{1.2}$$

$$\frac{dh}{dt} = \frac{h_\infty(V) - h}{\tau_h(V)} \tag{1.3}$$

$$\frac{dn}{dt} = \frac{n_\infty(V) - n}{\tau_n(V)} \tag{1.4}$$

$$\frac{dp}{dt} = \frac{p_\infty(V) - p}{\tau_p(V)} \tag{1.5}$$

$$\frac{dr_f}{dt} = \frac{r_{f\infty}(V) - r_f}{\tau_{rf}(V)} \tag{1.6}$$

$$\frac{dr_s}{dt} = \frac{r_{s\infty}(V) - r_s}{\tau_{rs}(V)} \tag{1.7}$$

The definition of the various $_x$ and τ_x can be found in the codes included in this report, and the original papers upon which this report are based. The physiological details can be found in the literature.

2. REDUCED MODEL

To continue are discussion, we reproduce Figure 1 of our topic paper below. Rotstein et al. note, from this figure, that $\tau_{r,f}$ and $\tau_{r,s}$ are much greater than all other τ_x in the regime of the STO, what they call the subthreshold interval

(STI), (compare Figure 2.1 with Figure 2.2). The main point upon which their work hinges is the idea that in the STI, I_{Na} and I_K are nearly inactive. From Figure 2.1, left column, one can also approximate that $m_\infty\ n_\infty^4 \approx 0$. Since τ_p is so small in this regime, it quickly evolves and so one can use the approximation $p \approx p_{\infty(v)}$, and from the previous assumptions, $m \approx 0, I_{Na} \approx 0$, and $I_K \approx 0$ due to their dependence on m and n.

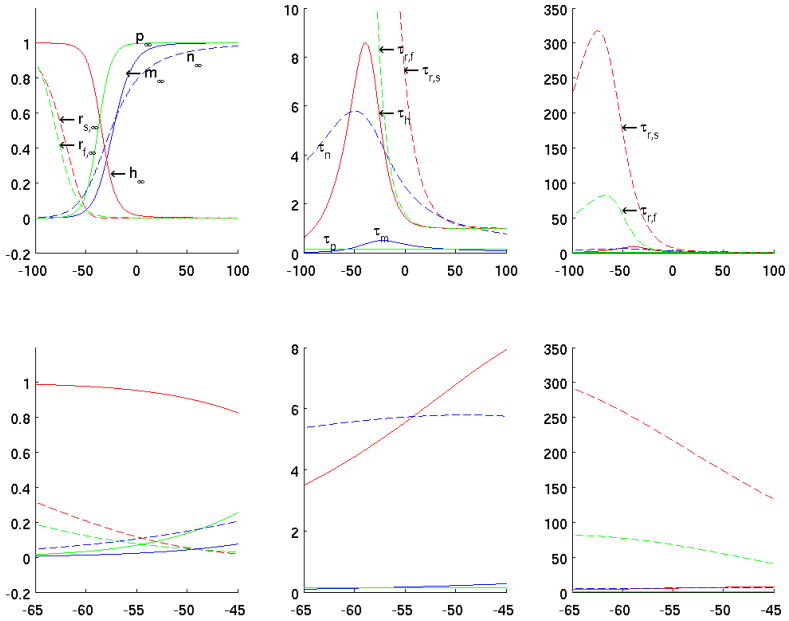


FIGURE 2.1. Recreation of Figure 1 of [5]. Ion channel dynamics for full seven dimensional SC model. Bottom row magnifies corresponding top row. First column shows activation and inactivation curves for the gating variables. Second column shows Voltage-dependent time scales. The third column shows the second column for a larger time interval. These time scales are key to the dynamical discussion of [5].

This sets up the reduced equations, given by,

$$C\frac{dV}{dt} = I_{app} - G_p p_\infty(V)(V - E_{Na}) - I_L - I_h \tag{2.1}$$

$$\frac{dr_f}{dt} = \frac{r_{f,\infty}(V) - r_f}{\tau_{r_f}(V)} \tag{2.2}$$

$$\frac{dr_s}{dt} = \frac{r_{s,\infty}(V) - r_s}{\tau_{r_s}(V)} \tag{2.3}$$

Next Rotstein et al. note that, as seen in Figure 2.1 at the third column, in the STI regime, $\tau_{r,s} >> \tau_{r,f}$ and so r_f can be taken as a slow system in the fast/slow decomposition. That is, we take r_s as a parameter rather than as an evolving variable. This brings us to our simplest possible system. In Figure 2.4, I use Maple to compute the Eigenvalues for the fixed point of the fast system as a function of

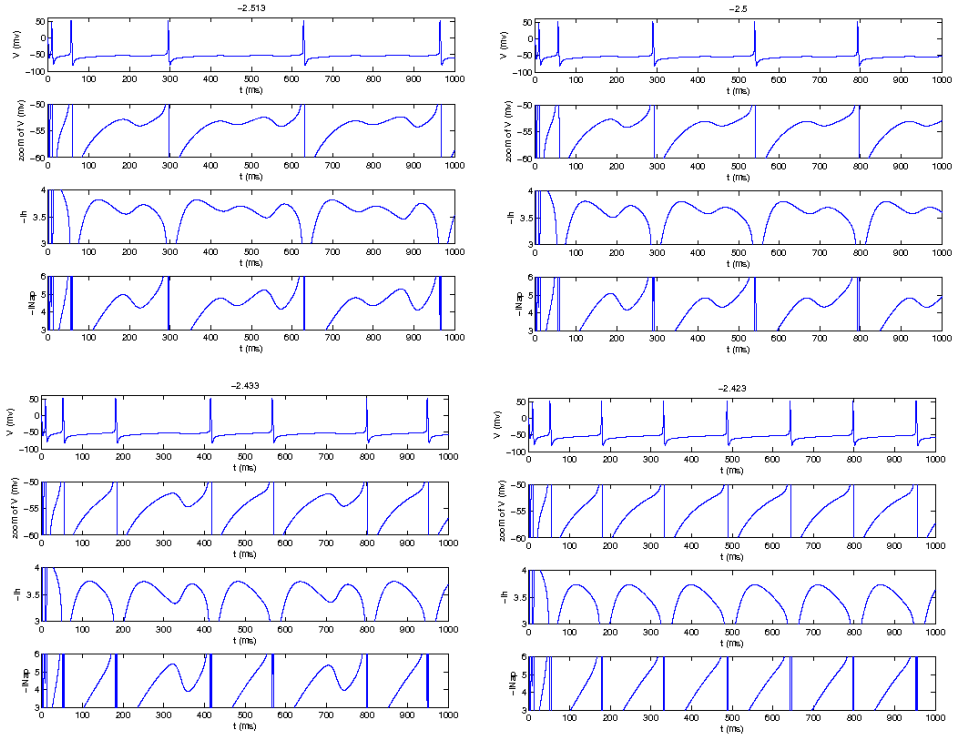


FIGURE 2.2. Approximate recreation of Figure 2 of [5] made using same MATLAB program created for Figure 1.2. Certain unknown initial conditions made an exact recreation unlikely. Full SC model with conditions given in program.

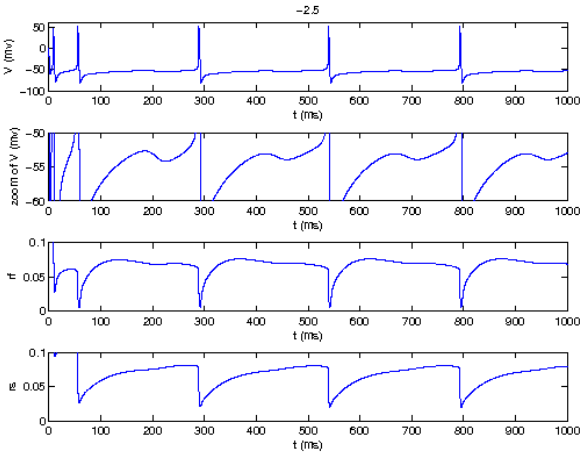


FIGURE 2.3. Approximate recreation of Figure 3 of [5] made using same MATLAB program created for Figure 1.2. Certain unknown initial conditions made an exact recreation unlikely. Full SC model with conditions given in program.

the variable of the slow system (r_s). This is done by calculating the nullclines for the two fast components of the reduced system, and solving for their intersection. Then, for each fixed point, we perform the typical localization, and then solve for the eigenvalues of the Jacobian. If the point is hyperbolic (nonzero real parts of eigenvalues) then the Hartman- Grobman theorem indicates that the results for the linearization of the system will match the results for the actual system (locally).

Center manifold theory can indicate the nature of the system at the point at which the fixed point becomes non-hyperbolic, i.e. when the real part of the eigenvalues crosses the imaginary axis. The result is the Andronov-Hopf bifurcation (sub-critical in this case).

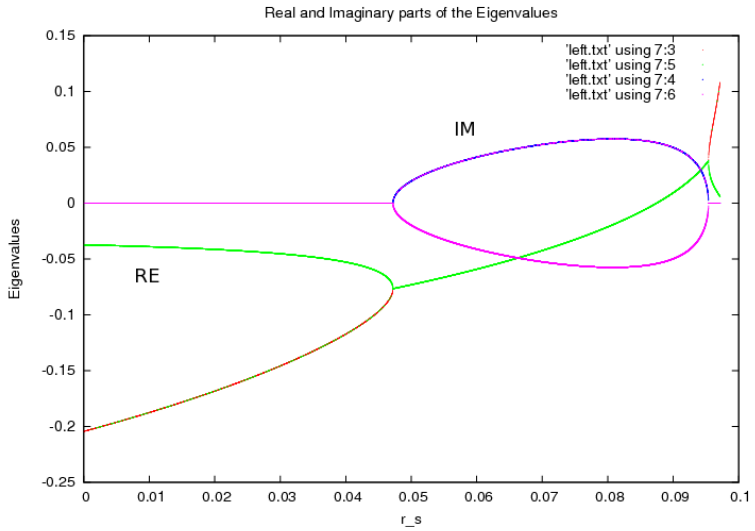


FIGURE 2.4. Eigenvalues for the fast component of the reduced SC model from [5] made using same MATLAB program created for Figure 1.2. Closer look at generating area of STO. The signature of a Hopf bifurcation is present.

The eigenvalue plot of Figure 2.4, created with the enclosed program **eigen-run.m**, suggests the following: In the figure we see the amplitude of the Eigenvalues as plotted for the r_s variable. from $r_s = 0.. \approx 0.047215$ we have two negative real eigenvalues, indicating a fixed point. At around that point, the eigenvalues collide and we now have two complex conjugate eigenvalues with negative real part. This indicates the presence of a spiral fixed point. Around $r_s \approx 0.087460$ the real part of the eigenvalues crosses the Imaginary axis, which is a signature for a Hopf Bifurcation (unstable central point). At $r_s \approx 0.095380$ the imaginary components of the eigenvalues collide and the fixed point becomes simply unstable (non-spiral) and all trajectories should avoid this fixed point. In Figure 2.5, detailed further in Figures 2.6 and 2.7, we see the dynamics indicated by the eigenvalues in the phase plane.

Obviously, the results of the analysis for the fast part of the system will not accurately describe the overall system; it does, however, give a strong indication of the dynamics that drive the STO. We now turn our attention to an analysis of the full reduced system.

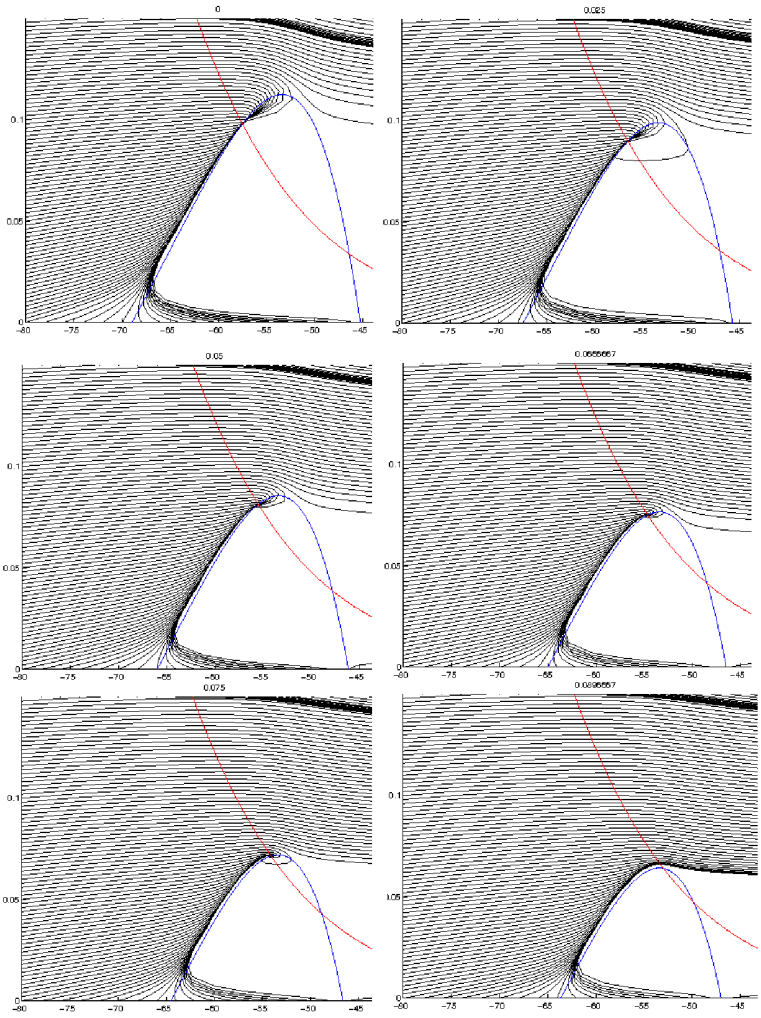


FIGURE 2.5. Approximate recreation of upper panel Figure 5 of [5] made using same MATLAB program created for Figure 1.2. Values of rs scan, from left to right and top to bottom: 0, 0.025, 0.050, 0.075, 0.0666..., and 0.089667.

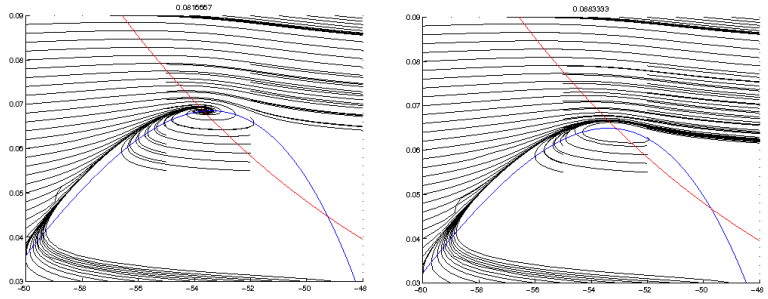


FIGURE 2.6. Approximate recreation of of Figure 5 of [5] made using same MATLAB program created for Figure 1.2. $r_s = 0.081667, 0.88333$, before and after the Hopf bifurcation indicated by the eigenvalues shown in Figure 2.4.

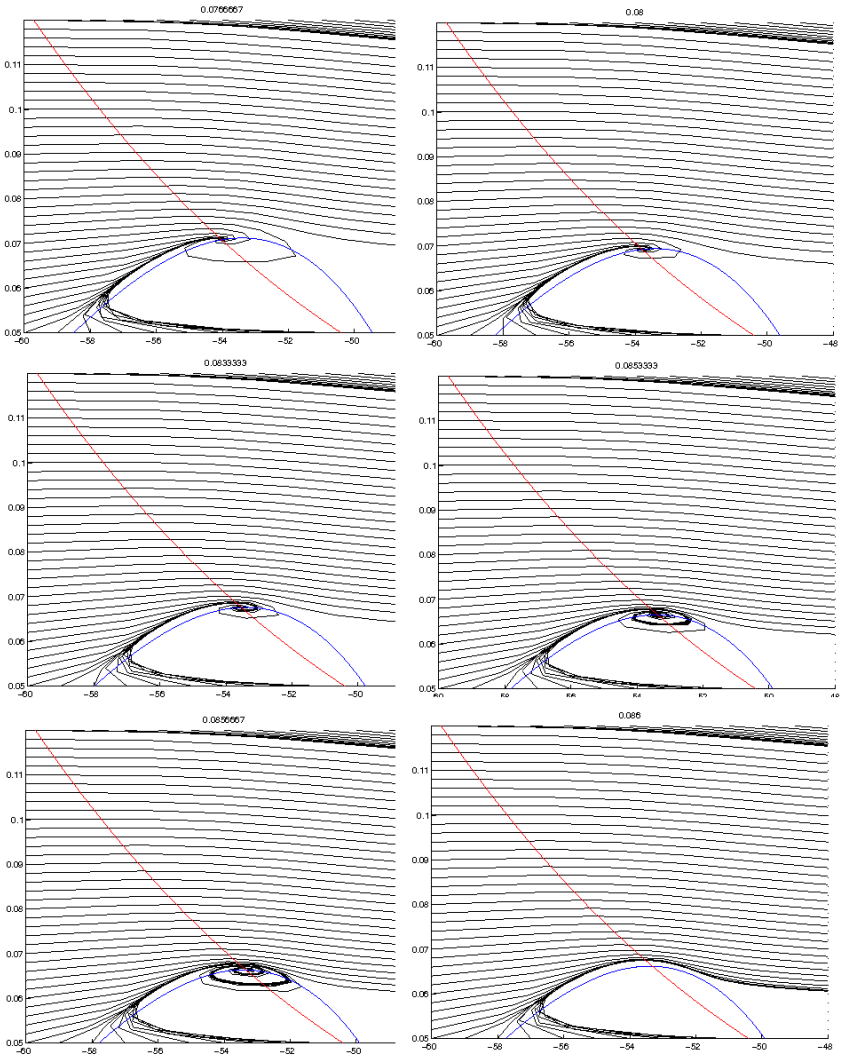


FIGURE 2.7. Approximate recreation of lower panel of Figure 5 of [5] made using same MATLAB program created for Figure 1.2. Values of rs scan, from left to right and top to bottom: 0.0766, 0.08, 0.0833, 0.0853, 0.0856, and 0.086.

3. Modeling of full reduced SC system

I wrote the enclosed C++ code to model the full three dimensional system using Runge-Kutta IV method. In the three dimensional system, the STO motion never settles to a fixed point; after a number of oscillations, the trajectory escapes (spike). The reduced model has no mechanism for return. However, following Rotstein et al., we introduce an artificial "reset" mechanism, i.e. the following line of code:

```
if(y[1]>-30){y[1]=-80; y[2]=0; y[3]=0; flag=0;}
```

This is justified in the article by the fact that under the full model, r_f and r_h reset to near zero after each action potential (see Figure 2.3). The result of our simulation of their simulation can be found in 3.1, with some additional detail.

After a certain threshold of applied current, the subthreshold oscillations can be seen to grow more prominent; we demonstrate this evolution in Figure 3.2.

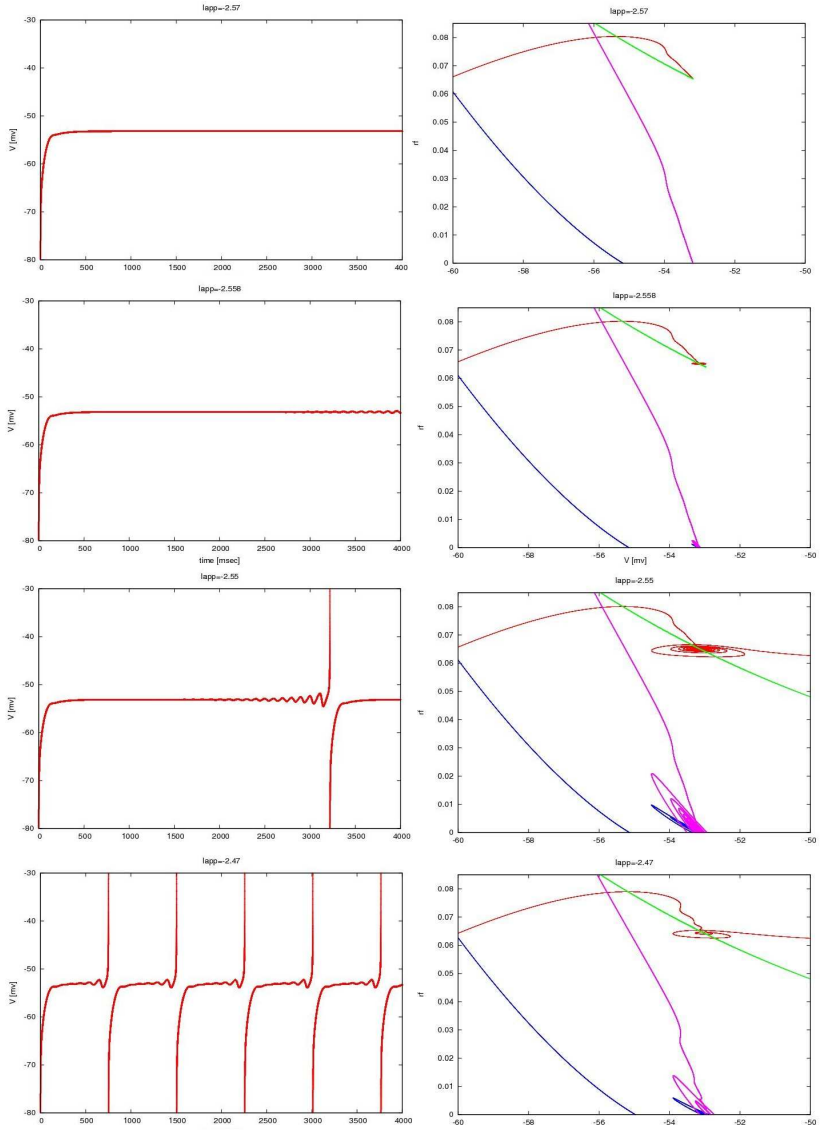


FIGURE 3.1. Approximate recreation of lower panel of Figure 6 of [5] made using same MATLAB program created for Figure 1.2. Values of I_{app} varied (from top to bottom) as -2.57, -2.558, -2.55, -2.47. The right panel is a bit abstract: the green line is the static r_f nullcline. The red line is the trajectory with initial conditions $(V, r_f, r_s) = (-80, 0, 0)$. The blue line represents the difference between the V and r_f nullcline for the given value of V, r_f, r_s that the trajectory has at the given V. The Purple line represents the difference between the V and r_s nullcline for the values of V, r_f, r_s that the trajectory has at the given V. Since the system is allowed to evolve in all three coordinates freely, no static nullclines for V can be drawn in this diagram. The "reset" in the left column is artificial, see text for more details.

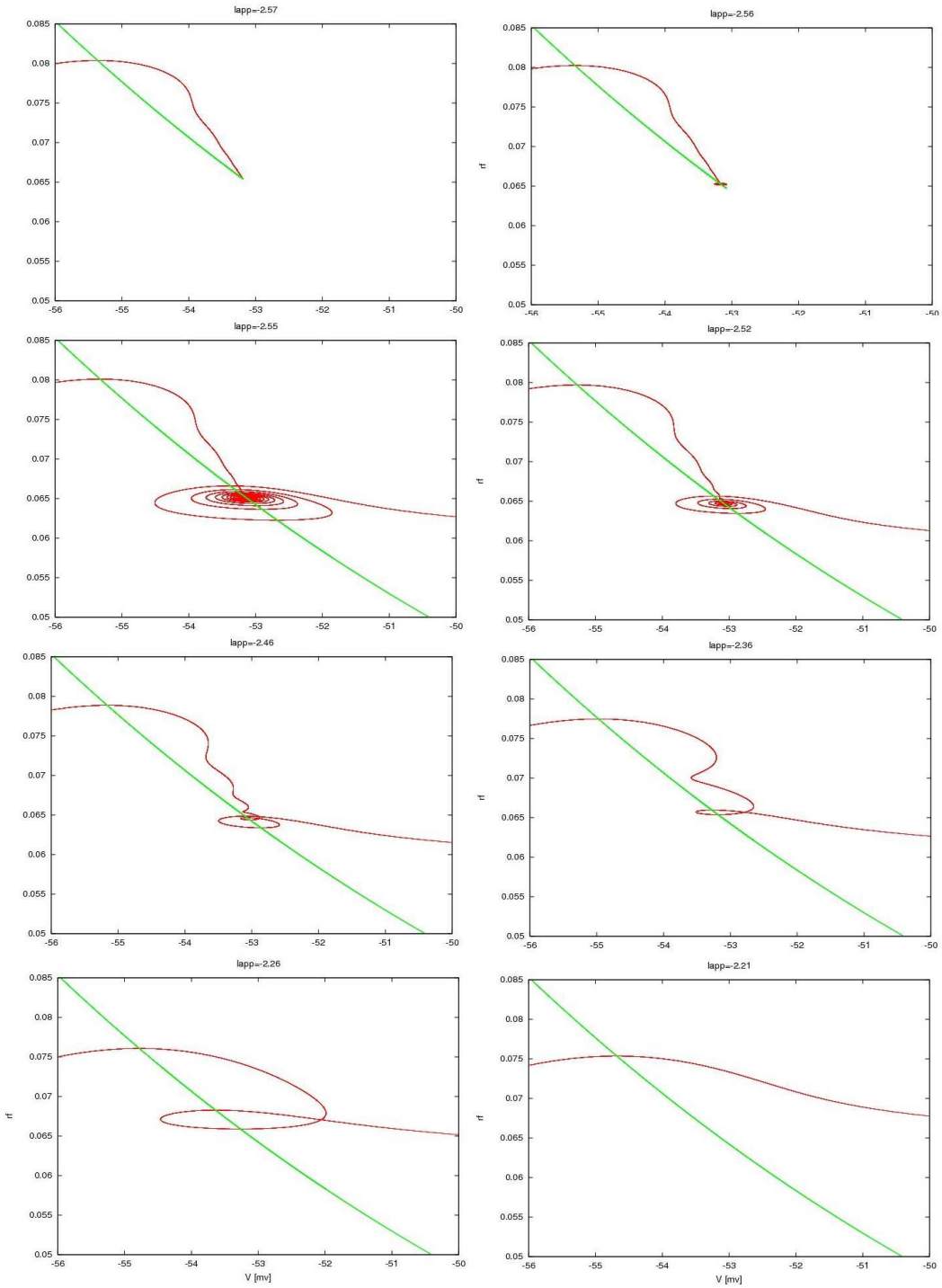


FIGURE 3.2. Approximate recreation of Figure 6 of [5] made using same MATLAB program created for Figure 1.2. Closer look at generating area of STO.

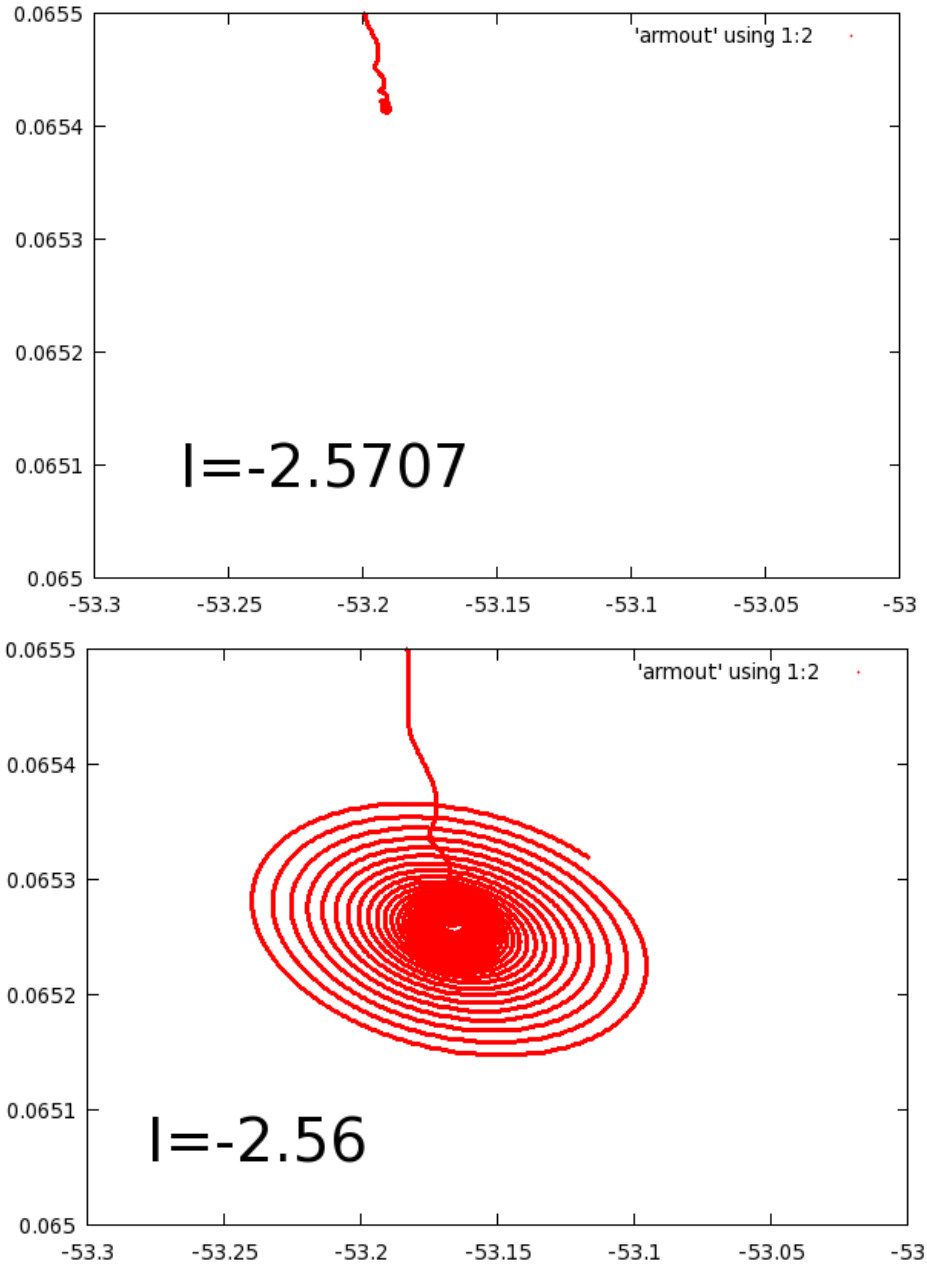


FIGURE 3.3. Close up of transitional area for foll reduced SC model.

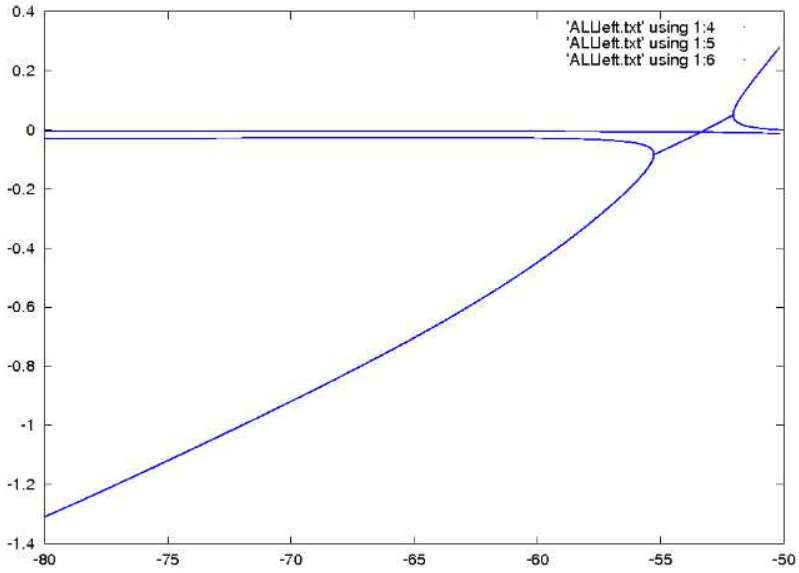


FIGURE 3.4. Eigenvalues for the fast component of the reduced SC model from [5] made using same MATLAB program created for Figure 1.2. Closer look at generating area of STO. This figure and next are recreations of Figure 12 of [5].

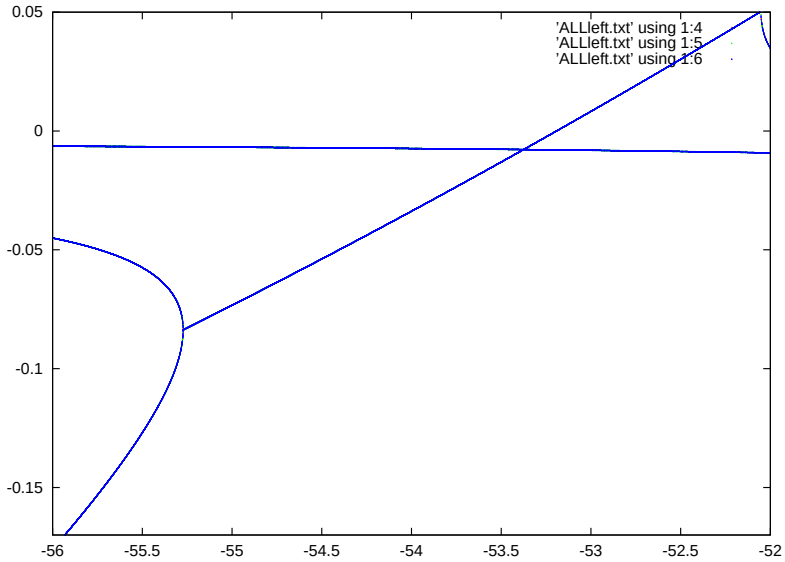
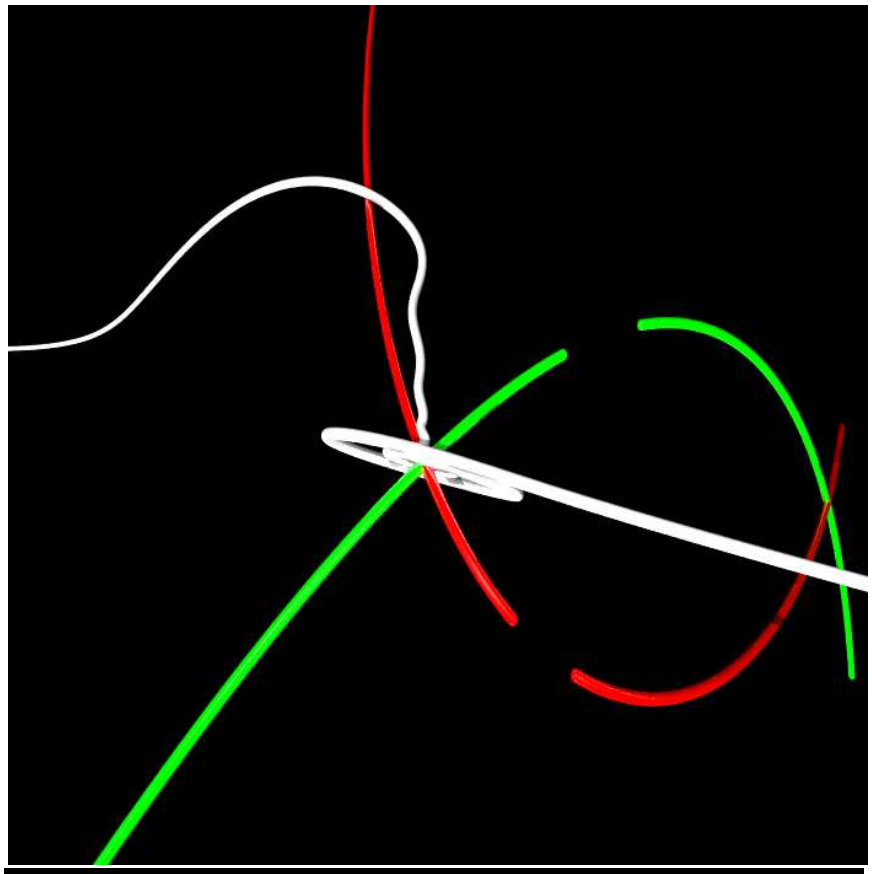
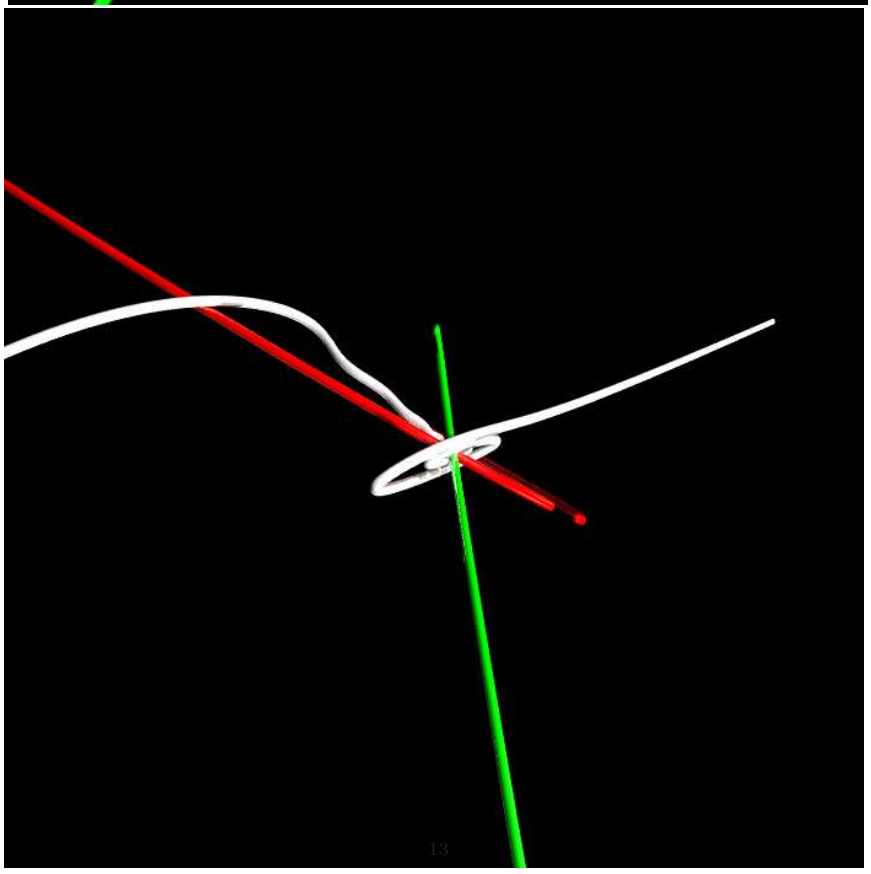
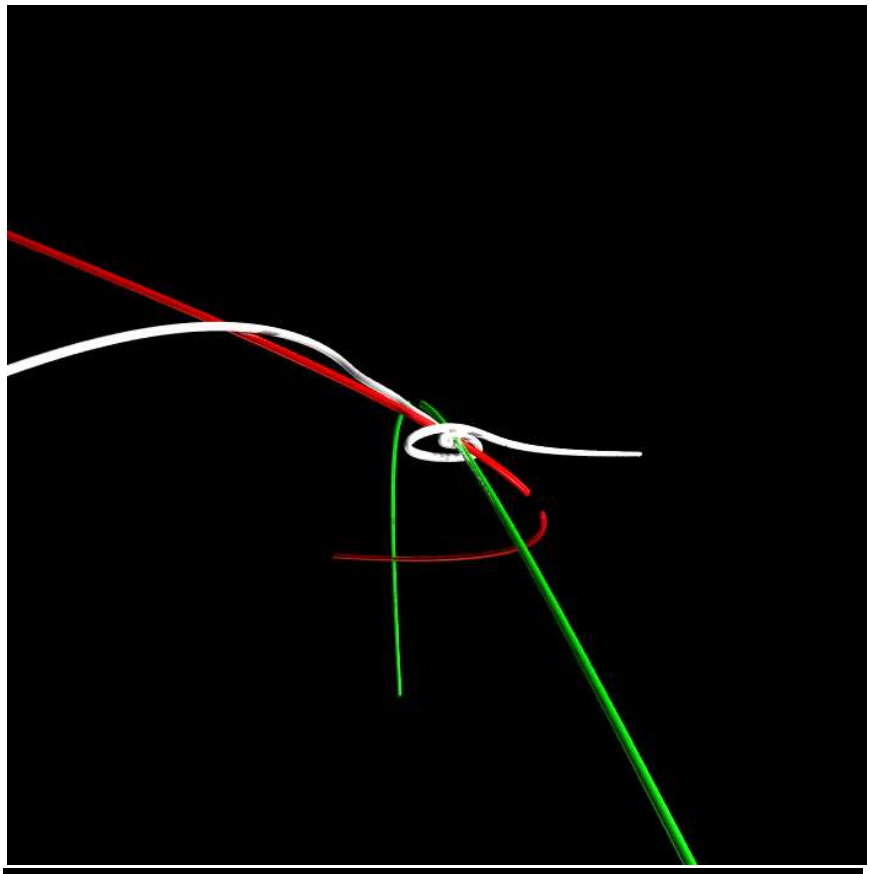
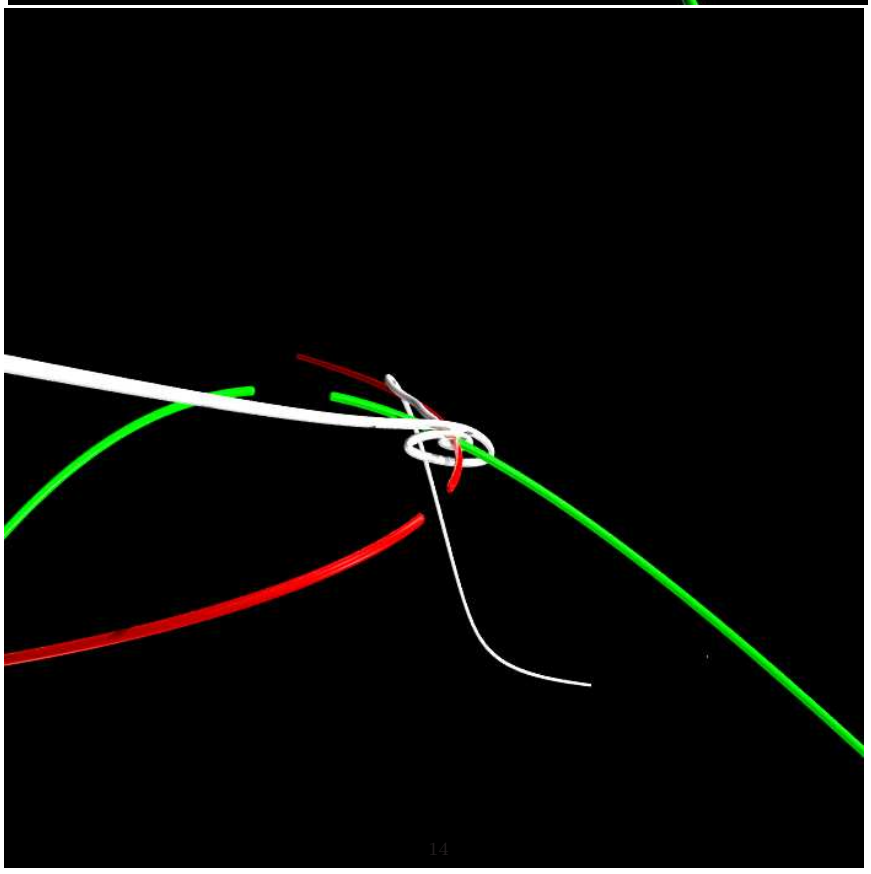


FIGURE 3.5. Zoom in of eigenvalues for the fast component of the reduced SC model from [5] made using same MATLAB program created for Figure 1.2. Closer look at generating area of STO. The signature of a Hopf bifurcation is present (imaginary values not shown).









References

[1] ALONSO, A. & KHLER, C. (1984) *A study of the reciprocal connections between the septum and the entorhinal area using anterograde and retrograde axonal transport methods in the rat brain.* J. Compar. Neurol. 225: 327343.

[2] JONES, R.S.G. (1993) *Entorhinal-hippocampal connections: A speculative view of their function.* Trends Neurosci. 16: 5864.

[3] NIH Press Release, *http://www.nimh.nih.gov/science-news/2007/cortex-area-thinner-in-youth-with-alzheimers-related-gene.shtml*

[4] Angel Alonso & Rodolfo R. Llinás (1989) *Subthreshold Na+-dependent theta-like rhythmicity in stellate cells of entorhinal cortex layer II* Nature 342, 175 - 177

[5] Horacio G. Rotstein, Tim Oppermann, John A. White, & Nancy Kopell (2006) *The dynamic structure underlying subthreshold oscillatory activity and the onset of spikes in a model of medial entorhinal cortex stellate cells* Journal of Computational Neuroscience 21:271-292

[6] Acker C, Kopell N, White J (2003) *Synchronization of strongly coupled excitatory neurons: Relating network behavior to biophysics.* Journal of Computational Neuroscience 15:71-90

[7] Eugen M. Izhikevich, (2007) *Dynamical Systems in Neuroscience*, The MIT Press, Cambridge

Acker_Fig2.m

```
function Acker(input)
global Iapp
Iapp=input;
T_MAX=1000;
step=0.05;
tspan=0:step:T_MAX;
x0=[-0.2828,  0.3208, 0.0513, 0.5841, 0.3, 0.3, 0.3]
[t,x]=ode15s(@Acker_function, tspan, x0);

v=x(:,1);
m=x(:,2);
h=x(:,3);
n=x(:,4);
p=x(:,5);
rf=x(:,6);
rs=x(:,7);

ENa=55;
EK=-90;
EL=-65;
Eh=-20;
GNa=52;
GK=11;
GL=0.5;
Gp=0.5;
Gh=1.5;
C=1;
Ih=Gh*(0.65.*rf + 0.35.*rs).*(v-Eh);
INap=Gp.*p.*(v-ENa);

figure(1)
subplot(4,1,1)
plot(t,v);
axis([0 T_MAX -100 60]);
xlabel ('t (ms)')
ylabel ('V (mv)')
title(Iapp);

subplot(4,1,2);
plot(t,v);
axis([0 T_MAX -60 -50]);
xlabel ('t (ms)')
ylabel ('zoom of V (mv)')

subplot(4,1,3);
plot(t,-Ih);
axis([0 T_MAX 3 4]);
xlabel ('t (ms)')
ylabel ('-Ih')

subplot(4,1,4);
```

```
plot(t,-INap);
axis([0 T_MAX 3 6]);
xlabel ('t (ms)')
ylabel ('-INap')
print -dpng 'acker1'
```

Acker_function (included inside the above and below Acker variations, this is the kernel of the main SC model)

```
function xdot = Acker_function(t,x)
global Iapp
v=x(1);
m=x(2);
h=x(3);
n=x(4);
p=x(5);
rf=x(6);
rs=x(7);
ENa=55;
EK=-90;
EL=-65;
Eh=-20;
GNa=52;
GK=11;
GL=0.5;
Gp=0.5;
Gh=1.5;
C=1;
am=-0.1*(v+23)/(exp(-0.1*(v+23))-1);
bm=4*exp(-(v+48)/18);
ah=0.07*exp(-(v+37)/20);
bh=1/(exp(-0.1*(v+7))+1);
an=-0.01*(v+27)/(exp(-0.1*(v+27))-1);
bn=0.125*exp(-(v+37)/80);
ap=1/(0.15*(1+exp(-(v+38)/6.5)));
bp=exp(-(v+38)/6.5)/(0.15*(1+exp(-(v+38)/6.5)));
rfi=1/(1+exp((v+79.2)/9.78));
trf=0.51/(exp((v-1.7)/10) + exp(-(v+340)/52)) + 1;
rsi=1/(1+exp((v+2.83)/15.9))^58;
trs=5.6/(exp((v-1.7)/14) + exp(-(v+260)/43)) + 1;
tp=0.15;
pinf=1/(1+exp(-(v+38)/6.5));
mi=am/(am+bm);
tm = 1/(am+bm);
dm=(mi -m)/tm;
hi=ah/(ah+bh);
th = 1/(ah+bh);
dh=(hi -h)/th;
ni=an/(an+bn);
tn = 1/(an+bn);
dn=(ni -n)/tn;
pi=ap/(ap+bp);
```

```
tp = 1/(ap+bp);
dp=(pi -p)/tp;
drf=(rfi -rf)/trf;
drs=(rsi -rs)/trs;
dv=Iapp - GNa*m^3*h*(v-ENa) - GK*n^4*(v-EK) - GL*(v-EL)
                          - Gh*(0.65*rf + 0.35*rs)*(v-Eh) - Gp*p*(v-ENa);
xdot=[dv dm dh dn dp drf drs]';
```

Acker_Fig1.m

```
function Acker(input)

global Iapp
Iapp=input;

T_MAX=1000;
step=0.05;
tspan=0:step:T_MAX;
v=-100:0.05:100;
am=-0.1.*(v+23)./(exp(-0.1.*(v+23))-1);
bm=4.*exp(-1.*(v+48)/18);
tm = 1./(am+bm);
ah=0.07.*exp(-1.*(v+37)/20);
bh=1./(exp(-0.1.*(v+7))+1);
th = 1./(ah+bh);
mi=am./(am+bm);
hi=ah./(ah+bh);

an=-0.01.*(v+27)./(exp(-0.1.*(v+27))-1);
bn=0.125.*exp(-(v+37)/80);
tn = 1./(an+bn);
ap=1./(0.15*(1+exp(-1.*(v+38)/6.5)));
bp=exp(-(v+38)/6.5)./(0.15*(1+exp(-1.*(v+38)/6.5)));
tp = 1./(ap+bp);
rfi=1./(1+exp((v+79.2)/9.78));
trf=0.51./(exp((v-1.7)/10) + exp(-1.*(v+340)/52)) + 1;
rsi=1./(1+exp((v+2.83)/15.9)).^58;
trs=5.6./(exp((v-1.7)/14) + exp(-1.*(v+260)/43)) + 1;
tp=0.15;
pinf=1./(1+exp(-(v+38)/6.5));
ni=an./(an+bn);
tn = 1./(an+bn);
ni=an./(an+bn);

miX=(-0.1*(0+23)/(exp(-0.1*(0+23))-1))/( -0.1*(0+23)/(exp(-0.1*(0+23))-1)
                                                + 4*exp(-(0+48)/18));
hiX=( 0.07*exp(-(-20+37)/20) ) / ( 0.07*exp(-(-20+37)/20)
                                          +  1/(exp(-0.1*(-20+7))+1)   );
pinfX=1/(1+exp(-(0+38)/6.5));
niX= (   -0.01.*(60+27)/(exp(-0.1*(60+27))-1)   ) / (    -0.01*(60+27)
                  /(exp(-0.1*(60+27))-1)    +    0.125*exp(-(60+37)/80) );

figure(1)
```

```
hold on

subplot(2,3,1)
hold on
axis([-100 100 -0.2 1.2]);
plot(v,mi, '-b');
text(v(:,1800),mi(:,1800),'\leftarrow m_\infty','HorizontalAlignment','left')
plot(v,hi, '-r');
text(v(:,1500),hi(:,1500),'\leftarrow h_\infty','HorizontalAlignment','left')
plot(v,pinf, '-g');
text(v(:,2000),pinf(:,2000),'p_\infty','HorizontalAlignment','left')
plot(v,ni, '--b');
text(v(:,3000),ni(:,3000)-0.07,'n_\infty','HorizontalAlignment','left')
plot(v,rsi, '--r');
text(v(:,500),rsi(:,500),'\leftarrow r_{s,\infty}','HorizontalAlignment','left')
plot(v,rfi, '--g');
text(v(:,500),rfi(:,500),'\leftarrow r_{f,\infty}','HorizontalAlignment','left')

subplot(2,3,4)

hold on
axis([-65 -45 -0.2 1.2]);
plot(v,mi, '-b');
plot(v,hi, '-r');
plot(v,pinf, '-g');
plot(v,ni, '--b');
plot(v,rsi, '--r');
plot(v,rfi, '--g');


subplot(2,3,2)
hold on
axis([-100 100 0 10]);
plot(v,tm, '-b');
text(v(:,1700),tm(:,1700)+0.4,'\tau_m','HorizontalAlignment','right')
plot(v,th, '-r');
text(v(:,1500),th(:,1500),'\leftarrow \tau_h','HorizontalAlignment','left')
plot(v,tp, '-g');
text(-80,tp+0.4,'\tau_p','HorizontalAlignment','left')
plot(v,tn, '--b');
text(v(:,300),tn(:,300)-0.07,'\tau_n','HorizontalAlignment','left')
plot(v,trs, '--r');
text(v(:,2000),trs(:,2000),'\leftarrow \tau_{r,s}','HorizontalAlignment','left')
plot(v,trf, '--g');
text(v(:,1500),trf(:,1500),'\leftarrow \tau_{r,f}','HorizontalAlignment','left')

subplot(2,3,5)

hold on
axis([-65 -45 0 8]);
plot(v,tm, '-b');
plot(v,th, '-r');
```

```
plot(v,tp, '-g');
plot(v,tn, '--b');
plot(v,trs, '--r');
plot(v,trf, '--g');

subplot(2,3,3)
hold on
axis([-100 100 0 350]);
plot(v,tm, '-b');

plot(v,th, '-r');
plot(v,tp, '-g');
plot(v,tn, '--b');
plot(v,trs, '--r');
text(v(:,1000),trs(:,1000),'\leftarrow \tau_{r,s}','HorizontalAlignment','left')
plot(v,trf, '--g');
text(v(:,1000),trf(:,1000),'\leftarrow \tau_{r,f}','HorizontalAlignment','left')

subplot(2,3,6)

hold on
axis([-65 -45 0 350]);
plot(v,tm, '-b');
plot(v,th, '-r');
plot(v,tp, '-g');
plot(v,tn, '--b');
plot(v,trs, '--r');
plot(v,trf, '--g');
```

Acker_FIG5.m: MATLAB code written to recreate Figure 5 from main journal article.

```
function ARM(input)
global rsNULL
rsNULL=input;
Iapp=-2.5;
ENa=55;
EK=-90;
EL=-65;
Eh=-20;
GNa=52;
GK=11;
GL=0.5;
Gp=0.5;
Gh=1.5;
C=1;
tspan=[0 350];
figure(2)
axis([-80 -40 0 0.15])
hold on
for i=-80:2:-40
x0=[i; 0; rsNULL];
```

```
[t,x]=ode23(@AckerRM_function, tspan, x0);
vv=x(:,1);
nn=x(:,2);
plot(vv,nn, '-black')
x0=[i; 0.15; rsNULL];
[t,x]=ode23(@AckerRM_function, tspan, x0);
vv=x(:,1);
nn=x(:,2);
plot(vv,nn, '-black')
end
for j=0:0.002:0.15
x0=[-80; j; rsNULL];
[t,x]=ode23(@AckerRM_function, tspan, x0);
vv=x(:,1);
nn=x(:,2);
plot(vv,nn, '-black')
x0=[-40; j; rsNULL];
[t,x]=ode23(@AckerRM_function, tspan, x0);
vv=x(:,1);
nn=x(:,2);
plot(vv,nn, '-black')
end
v=-80:0.1:-40;
pinf=1./(1+exp(-1.*(v+38)/6.5));
rfNULL1=(Iapp - GL.*(v-EL) - Gh*(0.35*rsNULL).*(v-Eh)
                      - Gp.*pinf.*(v-ENa))./(Gh*0.65.*(v-Eh));
rfNULL2=1./(1+exp((v+79.2)/9.78));
plot(v,rfNULL1, '-b');
plot(v,rfNULL2, '-r');
title(rsNULL);
print -dpng 'acker1'

function xdot = AckerRM_function(t,x)

global rsNULL
Iapp=-2.5;

v=x(1);
rf=x(2);
rs=x(3);

ENa=55;
EK=-90;
EL=-65;
Eh=-20;
GNa=52;
GK=11;
GL=0.5;
Gp=0.5;
Gh=1.5;
C=1;
```

```
am=-0.1*(v+23)/(exp(-0.1*(v+23))-1);
bm=4*exp(-(v+48)/18);
ah=0.07*exp(-(v+37)/20);
bh=1/(exp(-0.1*(v+7))+1);
an=-0.01*(v+27)/(exp(-0.1*(v+27))-1);
bn=0.125*exp(-(v+37)/80);
ap=1/(0.15*(1+exp(-(v+38)/6.5)));
bp=exp(-(v+38)/6.5)/(0.15*(1+exp(-(v+38)/6.5)));
rfi=1/(1+exp((v+79.2)/9.78));
trf=0.51/(exp((v-1.7)/10) + exp(-(v+340)/52)) + 1;
rsi=1/(1+exp((v+2.83)/15.9))^58;
trs=5.6/(exp((v-1.7)/14) + exp(-(v+260)/43)) + 1;
tp=0.15;
pinf=1/(1+exp(-(v+38)/6.5));
drf=(rfi -rf)/trf;
drs=0;
dv=Iapp - GL*(v-EL) - Gh*(0.65*rf + 0.35*rs)*(v-Eh) - Gp*pinf*(v-ENa);
xdot=[dv drf drs]';
```

eigenrun.m: This is a Maple program designed to solve for the fast part of the reduced SC system's eigenvalues.

```
restart: with(linalg): with(LinearAlgebra): Digits := 13:
fa := fopen("left.txt", WRITE):
fb :=fopen("right.txt",WRITE);
ENa := 55: EK := -90:
EL := -65: Eh := -20:
GNa := 52: GK := 11:
GL := .5: Gp := .5:
Gh := 1.5: C := 1:
Iapp := -2.5:

for rsN from 0 by 0.00001 to 0.1 do
pinf := proc (v) options operator, arrow; 1/(1+exp((-1)*(v+38)/6.5)) end proc:
rfNULL1 := (Iapp-GL*(v-EL)-.35*Gh*rsN*(v-Eh)-Gp*pinf(v)*(v-ENa))/(.65*Gh*(v-Eh)):
rfNULL2 := 1/(1+exp((v+79.2)/(9.78))):
vFIX1a := fsolve(rfNULL1 = rfNULL2, v = -60 .. -51):
rfFIX1a := 1/(1+exp((vFIX1a+79.2)/(9.78))):
vFIX2a := fsolve(rfNULL1 = rfNULL2, v = -51 .. -40):
rfFIX2a := 1/(1+exp((vFIX2a+79.2)/(9.78))):
rfi := 1/(1+exp((v+79.2)/(9.78))):
trf := .51/(exp((v-1.7)*1/10)+exp(-(v+340)*1/52))+1:
rsi := 1/(1+exp((v+2.83)/(15.9)))^58:
trs := 5.6/(exp((v-1.7)*1/14)+exp(-(v+260)*1/43))+1:
drf := proc (v, rf) options operator, arrow; (rfi-rf)/trf end proc:
dv := proc (v, rf) options operator, arrow; Iapp-GL*(v-EL)
               -Gh*(.65*rf+.35*rsN)*(v-Eh)-Gp*pinf(v)*(v-ENa) end proc;
ul := diff(dv(v, rf), v): f1 := unapply(ul, v, rf):
ur := diff(dv(v, rf), rf): f2 := unapply(ur, v, rf):
bl := diff(drf(v, rf), v): f3 := unapply(bl, v, rf):
br := diff(drf(v, rf), rf): f4 := unapply(br, v, rf):
A := matrix([[f1(vFIX1a, rfFIX1a), f2(vFIX1a, rfFIX1a)],
```

```
            [f3(vFIX1a, rfFIX1a), f4(vFIX1a, rfFIX1a)]]):
eigens := {eigenvalues(A)}:
eigen1 := op(1, eigens); eigen2 := op(2, eigens):
B := matrix([[f1(vFIX2a, rfFIX2a), f2(vFIX2a, rfFIX2a)],
            [f3(vFIX2a, rfFIX2a), f4(vFIX2a, rfFIX2a)]]):
eigens2 := {eigenvalues(B)}:
eigen21 := op(1, eigens2); eigen22 := op(2, eigens2):
fprintf(fa, "%f %f %f %f %f %f %f \n", vFIX1a, rfFIX1a, Re(eigen1), Im(eigen1),
                                                    Re(eigen2), Im(eigen2),rsN):
fprintf(fb, "%f %f %f %f %f %f %f \n", vFIX2a, rfFIX2a, Re(eigen21), Im(eigen21),
                                                  Re(eigen22), Im(eigen22),rsN):
end do:
```

eigenall.m: This is a Maple program designed to solve for the reduced SC system's eigenvalues.

```
restart: with(linalg): with(LinearAlgebra): Digits := 15:
fa := fopen("ALLleft.txt", WRITE):
ENa := 55: EK := -90:
EL := -65: Eh := -20:
GNa := 52: GK := 11:
GL := .5: Gp := .5:
Gh := 1.5: C := 1:


for Iapp from -58.1 by 0.0001 to 10 do
pinf := proc (v) options operator, arrow; 1/(1+exp((-1)*(v+38)/6.5)) end proc:
rfi := 1/(1+exp((v+79.2)/(9.78))):
trf := .51/(exp((v-1.7)*1/10)+exp(-(v+340)*1/52))+1:
rsi := 1/(1+exp((v+2.83)/(15.9)))^58:
trs := 5.6/(exp((v-1.7)*1/14)+exp(-(v+260)*1/43))+1:
drf := (rfi-rf)/trf:
dv := Iapp-GL*(v-EL)-Gh*(.65*rf+.35*rs)*(v-Eh)-Gp*pinf(v)*(v-ENa);
drs := (rsi-rs)/trs:
f1 := unapply(dv, (v, rf, rs)):
f2 := unapply(drf, (v, rf, rs)):
f3 := unapply(drs, (v, rf, rs)):
fixed := fsolve({f1(v, rf, rs) = 0, f2(v, rf, rs) = 0, f3(v, rf, rs) = 0},
                                        {v=-80..-50, rf = 0 .. 1, rs = 0 .. 1}):
assign(fixed):
vFIXED := v:
rfFIXED := rf:
rsFIXED := rs:
unassign('v', 'rs', 'rf'):
vec := Vector(3, [f1(v, rf, rs), f2(v, rf, rs), f3(v, rf, rs)]);
A := evalf(subs({v = vFIXED, rf = rfFIXED, rs = rsFIXED}, jacobian(vec, [v, rf, rs]))):
ev := eigenvectors(A):
fprintf(fa, "%f %f %f %f %f %f %f %f %f %f \n", vFIXED, rfFIXED, rsFIXED, Re(ev[1][1]),
             Re(ev[2][1]), Re(ev[3][1]), Im(ev[1][1]), Im(ev[2][1]), Im(ev[3][1]),Iapp ):
end do:
```

all.sh: This is a typical shell script used to create frames for films that run the above MATLAB programs.

```
#!/bin/sh

j=1000
jj=3000
counter=-2570
while [ $counter -lt -2200 ]
do
i=‘echo $counter 1000 | awk ’{ print $1/$2}’‘
echo $i
./ai $i
./null $i

echo "set terminal postscript color" > gnu_pre.dat
echo "set output ’$j.ps’" >> gnu_pre.dat
echo "set title ’Iapp=$i’" >> gnu_pre.dat

echo "set terminal postscript color" > gnu_pre2.dat
echo "set output ’$jj.ps’" >> gnu_pre2.dat
echo "set title ’Iapp=$i’" >> gnu_pre2.dat

cat gnu_pre.dat gnu.dat > gnuout.dat
cat gnu_pre2.dat gnu2.dat > gnuout2.dat

gnuplot gnuout.dat
gnuplot gnuout2.dat

mogrify -format jpg $j.ps
mogrify -format jpg $jj.ps
mogrify -rotate 90 $j.jpg
mogrify -rotate 90 $jj.jpg

rm $j.ps
rm $jj.ps


let j=j+1
let jj=jj+1
let counter=counter+1

done
```

ARM_input.c: This C code was used, in conjunction with the above shell script, to create a series of images for the full reduced SC system.

```
//An ARM iterator
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
#include "nr.h"
#include "nrutil.h"
#include "nrutil.c"
//#include "rk4.c"

void rk4(double a, double y[], double dydx[], int n, double x, double h, double yout[],
void (*derivs)(double,double, double [], double []))
{
int i;
double xh,hh,h6,*dym,*dyt,*yt;

dym=dvector(1,n);
dyt=dvector(1,n);
yt=dvector(1,n);
hh=h*0.5;
h6=h/6.0;
xh=x+hh;
for (i=1;i<=n;i++) yt[i]=y[i]+hh*dydx[i];
(*derivs)(a,xh,yt,dyt);
for (i=1;i<=n;i++) yt[i]=y[i]+hh*dyt[i];
(*derivs)(a,xh,yt,dym);
for (i=1;i<=n;i++) {
yt[i]=y[i]+h*dym[i];
dym[i] += dyt[i];
}
(*derivs)(a,x+h,yt,dyt);
for (i=1;i<=n;i++)
yout[i]=y[i]+h6*(dydx[i]+dyt[i]+2.0*dym[i]);
free_vector(yt,1,n);
free_vector(dyt,1,n);
free_vector(dym,1,n);
}


int main(int argc, char *argv[]){

  int i1,i2,i3,j,n1,n,N;  double x,yold; double h=0.01; double *y;
double *dydx; double *yout;
n=3;

 int flag=100;

 double a=atof(argv[1]);
 //  printf("%f \n", a);
FILE* fd = fopen("armout", "w");
FILE* fe = fopen("armout2", "w");

y = dvector(1,n);  dydx = dvector(1,n); yout = dvector(1,n);

 y[1]=-80;
 y[2]=0.0;
 y[3]=0.0;
```

```
 x=0.0;
 for(i2=1; i2<=400000; i2++){
    x = x + h;
    derivs(a,x,y,dydx);
    rk4(a,y,dydx,n,x,h,yout,derivs);
    for(j=1; j<=n; j++){y[j]=yout[j];}
    //    yold = y[1];
    // if(y[1]<-40){

    ///GRAB THE NULLCLINES

double Iapp=a;
//printf("%f \n", Iapp);

double v=y[1];
double rf=y[2];
double rs=y[3];

double ENa=55;
double EK=-90;
double EL=-65;
double Eh=-20;
double GNa=52;
double GK=11;
double GL=0.5;
double Gp=0.5;
double Gh=1.5;
double C=1;

double  am=-0.1*(v+23)/(exp(-0.1*(v+23))-1);
double  bm=4*exp(-(v+48)/18);
double  ah=0.07*exp(-(v+37)/20);
double  bh=1/(exp(-0.1*(v+7))+1);
double  an=-0.01*(v+27)/(exp(-0.1*(v+27))-1);
double  bn=0.125*exp(-(v+37)/80);
double  ap=1/(0.15*(1+exp(-(v+38)/6.5)));
double  bp=exp(-(v+38)/6.5)/(0.15*(1+exp(-(v+38)/6.5)));
double  rfi=1/(1+exp((v+79.2)/9.78));
double  trf=0.51/(exp((v-1.7)/10) + exp(-(v+340)/52)) + 1;
 double  rsi=pow(1/(1+exp((v+2.83)/15.9)),58);
double  trs=5.6/(exp((v-1.7)/14) + exp(-(v+260)/43)) + 1;
double  tp=0.15;
double  pinf=1/(1+exp(-(v+38)/6.5));

double rfNULL1=(Iapp - GL*(v-EL) - Gh*(0.35*rs)*(v-Eh) - Gp*pinf*(v-ENa))
                                                        /(Gh*0.65*(v-Eh));
double rfNULL2=1/(1+exp((v+79.2)/9.78));
double rsNULL1=(Iapp - GL*(v-EL) - Gh*(0.65*rf)*(v-Eh) - Gp*pinf*(v-ENa))
                                                        /(Gh*0.35*(v-Eh));
double rsNULL2=pow(1/(1+exp((v+2.83)/15.9)),58);
```

```
//double rfNULL3=pow(1/(1+exp((v+2.83)/15.9)),58);

////////////////////////////////////////////////////////////

 fprintf(fd,"%f %f %f %f \n", y[1], y[2], y[3], x);//}
 if(y[1]>-30){y[1]=-80; y[2]=0; y[3]=0; flag=0;}
 if(flag>10){ fprintf(fe,"%f %f %f %f %f %f %f \n", y[1], y[2], y[3],
                          x,rfNULL2, rfNULL2-rfNULL1,rsNULL2-rsNULL1);}
   }  //ends i2 loop
  printf("%f \n", y[1]);
}  //ends main

void derivs(double a, double x, double y[], double dydx[]){

double Iapp=a;
//printf("%f \n", Iapp);

double v=y[1];
double rf=y[2];
double rs=y[3];

double ENa=55;
double EK=-90;
double EL=-65;
double Eh=-20;
double GNa=52;
double GK=11;
double GL=0.5;
double Gp=0.5;
double Gh=1.5;
double C=1;

double  am=-0.1*(v+23)/(exp(-0.1*(v+23))-1);
double  bm=4*exp(-(v+48)/18);
double  ah=0.07*exp(-(v+37)/20);
double  bh=1/(exp(-0.1*(v+7))+1);
double  an=-0.01*(v+27)/(exp(-0.1*(v+27))-1);
double  bn=0.125*exp(-(v+37)/80);
double  ap=1/(0.15*(1+exp(-(v+38)/6.5)));
double  bp=exp(-(v+38)/6.5)/(0.15*(1+exp(-(v+38)/6.5)));
double  rfi=1/(1+exp((v+79.2)/9.78));
double  trf=0.51/(exp((v-1.7)/10) + exp(-(v+340)/52)) + 1;
 double  rsi=pow(1/(1+exp((v+2.83)/15.9)),58);
double  trs=5.6/(exp((v-1.7)/14) + exp(-(v+260)/43)) + 1;
double  tp=0.15;
double  pinf=1/(1+exp(-(v+38)/6.5));

dydx[1]=Iapp - GL*(y[1]-EL) - Gh*(0.65*y[2] + 0.35*y[3])*(y[1]-Eh) - Gp*pinf*(y[1]-ENa);
dydx[2]=(rfi -y[2])/trf;
dydx[3]=(rsi -y[3])/trs;
}
```