**Appendix A-Z**

What follows is a library of all of the scripts developed for the project completed above, and a reference to what portions of the project each of the codes was used for.

MLESIM (Q1 – Q4)

% MLESIM - simulate Morris-Lecar equations

% Needs a column vector pml of params:

% pml=[gca, gk, gl, vca, vk, vl, phi, v1, v2, v3, v4, v5, v6, C, vic, wic]'

% For meaning of parameters, see Rinzel and Ermentrout.

% and a column vector iext describing the external current

% iext = [iampl, tstart, tstop]'

% which are the current amplitude and the start and stop times.

% These are set up internally to the values for Fig. 7.1 of R&E if not supplied

% in the workspace.

% This always runs with i.c.s set to 0, just to give a spike.

% Parameters for Fig. 7.1. NOTE initial conditions are not included.

% NOTE VALUE OF phi CHANGED FROM FIG. 7.1 VALUE.

% gca, gk, gl, vca, vk, vl, phi, v1, v2,v3, v4,v5, v6, C,junk,junk

pml=[4.4, 8.0, 2, 120, -84, -60, 0.02, -1.2, 18, 2, 30, 2, 30, 20, 0, 0]';

% External current parameters.

% iext tstart tstop

iext = [0, 0, 0]';

figure(1); clf % Shouldn't be necessary but prevents an annoying bug

% in odeplot

% Store parameters for mlode:

setmleparms(pml, iext);

% Simulate for 100 ms from 0 initial conditions (should produce an AP)

tspan = [0; 100];

y0 = [0; 0];

% Show the state variables during the simulation, tell solver where to get

% the Jacobian.

options = odeset('OutputFcn', @odeplot, 'Jacobian', @mlodejac, ...

'Vectorized', 'on');

%options = odeset('Jacobian', @mlodejac, 'Vectorized', 'on');

% Do the simulation

[t,y] = ode15s(@mlode, tspan, y0, options);

% Print final value of state variables

current = y(end,:); fprintf('Final values: v=%g, w=%g\n',current);

% Replot the state variables so the W variable can be seen

[ax, h1, h2] = plotyy(t, y(:,1), t, y(:,2));

axes(ax(1)); ylabel('V, mV.')

axes(ax(2)); ylabel('W')

xlabel('Time, ms.');

v = -80:140/(length(t)-1):60;

% To make a phase plot

phase\_plane(t,v,y);

figure(2);clf

% locs1 = find (abs(null(1,:)-null(2,:))<0.005);

% thing = null(1,:)./null(2,:);

% locs2 = [];

% for i=2:length(thing)

% if (thing(i-1) < 1 && thing (i) > 1) || (thing(i-1) > 1 && thing(i) < 1)

% locs2 = [locs2, i];

% end

% end

% plot(VV(locs1), vnull(locs1), 'ro', VV(locs2), vnull(locs2), 'ko');

MLEC (Q5 – Q12)

% MLEC - simulate Morris-Lecar equations

% Needs a column vector pml of params:

% pml=[gca, gk, gl, vca, vk, vl, phi, v1, v2, v3, v4, v5, v6, C, ~, ~]'

% (For meaning of parameters, see Rinzel and Ermentrout, last two params are

% not used) and a column vector iext describing the external current

% iext = [iampl, tstart, tstop]'

% These will be set up internally to the values for R&E Fig. 7.1 if not supplied.

% NOTE VALUE OF phi CHANGED FROM FIG. 7.1 VALUE!

% if exist('pml')==0

pml=[4.4, 8.0, 2, 120, -84, -60, 0.04, -1.2, 18, 2, 30, 2, 30, 20, 0, 0]';

% fprintf('\*\*\*pml not supplied, set up internally.\*\*\*\n')

% end

if exist('iext')==0

iext = [0, 0, 0]';

fprintf('\*\*\*iext not supplied, set up internally.\*\*\*\n')

end

% Initialize display and set options for ode solvers.

figure(1); clf

options = odeset('OutputFcn', @odeplot, 'Jacobian', @mlodejac, ...

'Vectorized', 'on');

what = 'p';

while what~='q' & what~='Q'

tspan = [0, input('Enter stop time (ms, 100 is good) ')]';

if tspan(2)<=0; tspan(2) = 100; end

sy0s = input( ...

sprintf('Enter i.c.s (e.g. [v,w]=''%g %g''). <enter> to use 0s: ', ...

pml(15),pml(16)),'s');

if isempty(sy0s)

fprintf('\*\*\*Using default i.v.s (%g & %g).\*\*\*\n',pml(15:16))

y0=pml(15:16);

else

y0 = sscanf(sy0s, '%g %g', 2);

end

titl = sprintf('M-L eqns, vic=%g mV, hic=%g, iext=%g uA/cm^2.', ...

y0, iext(1));

figure(1); clf % to avoid a bug in odeplot

setmleparms(pml, iext); % Record parameters for mlode

[t,y] = ode15s(@mlode, tspan, y0, options); % Do the simulation

current = y(end,:);fprintf('Final values: v=%g, w=%g\n',current);

what = 'p';

while what=='p' | what=='P' | what=='h' | what=='H'

if what=='p' | what=='P'

clf

[ax, h1, h2] = plotyy(t, y(:,1), t, y(:,2));

axes(ax(1)); axis([0 tspan(2) -80 50]); ylabel('V, mV.')

axes(ax(2)); axis([0 tspan(2) 0 0.5]); ylabel('W')

xlabel('Time, ms.'); title(titl)

elseif what=='h' | what=='H'

figure(3)

vlim = [-80 60];

[y2, vnull, wnull] = makenulls(-80:1/(length(t)-1):60);

% fplot(vnull, vlim);

hold on

% fplot(wnull, vlim);

phi = [0.04 0.02 0.01];

for i=1:1

% pml(7) = phi(i);

% setmleparms(pml, iext);

% [t,y] = ode15s(@mlode, tspan, y0, options); % Do the simulation

plot(y(:,1), y(:,2))

end

xlabel('V, mV');ylabel('W')

axis([-80 50 0 0.5])

title(titl)

end

what = input('Again, Plot, pHase-plot, or Quit? ','s');

end

end

MLODE & MLODE\_reverse (Q2 – Q13)

function ydot = mlode(t, y)

% MLODE - ODE file for the Morris-Lecar Equations.

% Evaluates the derivative of the state vector for the Morris-Lecar

% equations with parameters pml, where pml is a column vector of params

% pml=[gca, gk, gl, vca, vk, vl, phi, v1, v2, v3, v4, v5, v6 C vic wic]'

% and iext is a column vector describing the external current

% iext = [iampl, tstart, tstop]'

% v5=v3 and v6=v4 are for tauw().

% Parameters are set through function SETMLEPARMS() only. mlode reads

% the parameters using GETMLEPARMS.

% ydot = mlode(t,y); returns dy/dt eval at t,y

% Note, also available:

% jac = mlodejac(t,y) returns the Jacobian at t,y

% Note mlode is vectorized, but mlodejac is not.

% Get parameters

[pml,iext] = getmleparms;

% Compute

ydot = zeros(2,1);

if t>=iext(2) & t<iext(3)

ydot(1) = (iext(1) - pml(1)\*minf(y(1),pml).\*(y(1)-pml(4)) - ...

pml(2)\*y(2).\*(y(1)-pml(5)) - pml(3)\*(y(1)-pml(6)))/pml(14);

else

ydot(1) = (-pml(1)\*minf(y(1),pml).\*(y(1)-pml(4)) - ...

pml(2)\*y(2).\*(y(1)-pml(5)) - pml(3)\*(y(1)-pml(6)))/pml(14);

end

ydot(2) = pml(7)\*(winf(y(1),pml)-y(2))./tauw(y(1),pml);

return

function ydot = mlode(t, y)

% MLODE - ODE file for the Morris-Lecar Equations.

% Evaluates the derivative of the state vector for the Morris-Lecar

% equations with parameters pml, where pml is a column vector of params

% pml=[gca, gk, gl, vca, vk, vl, phi, v1, v2, v3, v4, v5, v6 C vic wic]'

% and iext is a column vector describing the external current

% iext = [iampl, tstart, tstop]'

% v5=v3 and v6=v4 are for tauw().

% Parameters are set through function SETMLEPARMS() only. mlode reads

% the parameters using GETMLEPARMS.

% ydot = mlode(t,y); returns dy/dt eval at t,y

% Note, also available:

% jac = mlodejac(t,y) returns the Jacobian at t,y

% Note mlode is vectorized, but mlodejac is not.

% Get parameters

[pml,iext] = getmleparms;

% Compute

ydot = zeros(2,1);

if t>=iext(2) & t<iext(3)

ydot(1) = -(iext(1) - pml(1)\*minf(y(1),pml).\*(y(1)-pml(4)) - ...

pml(2)\*y(2).\*(y(1)-pml(5)) - pml(3)\*(y(1)-pml(6)))/pml(14);

else

ydot(1) = -(-pml(1)\*minf(y(1),pml).\*(y(1)-pml(4)) - ...

pml(2)\*y(2).\*(y(1)-pml(5)) - pml(3)\*(y(1)-pml(6)))/pml(14);

end

ydot(2) = - pml(7)\*(winf(y(1),pml)-y(2))./tauw(y(1),pml);

return

Q7

pml=[4.4, 8.0, 2, 120, -84, -60, 0.02, -1.2, 18, 2, 30, 2, 30, 20, 0, 0]';

figure(1); clf;

options = odeset('OutputFcn', @odeplot, 'Jacobian', @mlodejac, ...

'Vectorized', 'on');

tspan = [0 900];

iext = [86, 0, 900]';

setmleparms(pml, iext);

[y2, vnull, wnull] = makenulls(1);

subplot(122);

fplot(vnull, [-80 50]);

hold on;

fplot(wnull, [-80 50]);

legend('v nullcline', 'w nullcline');

titl = sprintf('M-L eqns, vic ranging, hic=0, iext=0 uA/cm^2.');

y0s = [-60.8554, 0.0149; -27.9524, 0.1195 ;-27.9, 0.17];

cols = ['k', 'b', 'r'];

for i=1:length(y0s)

y0 = y0s(i,:); figure(2);clf;

[t,y] = ode15s(@mlode, tspan, y0, options); % Do the simulation

figure(1)

hold on;

subplot(121);

plot(t, y(:,1), cols(i));

hold on;

ylabel('V, mV');xlabel('Time, ms.'); title(titl)

hold on;

subplot(122);

plot(y(:,1), y(:,2), cols(i))

xlabel('V, mV');ylabel('W');title(titl)

end

figure (1)

ylim([0 0.6]); xlim([-80 50]);

close figure 2

vv = @(v) vnull(v) - wnull(v);

e(1) = fzero(vv, -20);

e(2) = vnull(e(1));

jac = mlodejac(1, e);

eval = eig(jac)

Q10

pml=[4.4, 8.0, 2, 120, -84, -60, 0.02, -1.2, 18, 2, 30, 2, 30, 20, 0, 0]';

% figure(1); clf;

% options = odeset('OutputFcn', @odeplot, 'Jacobian', @mlodejac, ...

% 'Vectorized', 'on');

tspan = [0 500];

is = [10, 0 ,500];

for j = 1:500

if j >= 250

is(1) = is(1) + 0.02;

else

is(1) = is(1) + 0.3;

end

setmleparms(pml, is');

[~, vnull, wnull] = makenulls(1);

vv = @(v) vnull(v) - wnull(v);

% fplot(vnull, [-70 40]);

% hold on

% fplot(wnull, [-70 40]);

e(1) = fzero(vv, -20);

e(2) = vnull(e(1));

jac = mlodejac(1, e);

[a, b] = eig(jac);

evec{j} = a;

eval(j,:) = [b(1), b(4)];

if real(round(eval(j,:)\*100000)) == 0

current = is(1)

break;

end

end

eval;

% figure (1); clf;

% plot(real(eval), imag(eval))

% hold on

% plot(real(eval(end,:)), imag(eval(end,:)), 'ko')

iext = [80; 0; 500];

tspan = [0 500];

for j = 1:200

if j ~= 1

iext(1) = iext(1) + 0.1;

end

setmleparms(pml, iext);

[~, vnull, wnull] = makenulls(1);

vv = @(v) vnull(v)-wnull(v);

ex(1) = fzero(vv,-20);

ex(2) = vv(ex(1));

clf

[t,y] = ode15s(@mlode, tspan, ex);

n = 1;

for k = 1:length(y)-1

if y(k) <= 0 && y(k+1) > 0

pk(n) = t(k+1);

hold on;

plot(t(k+1), y(k+1), 'ko');

n = n+1;

end

end

rate(j) = 0;

if n > 2

rate(j) = (n-2)/(pk(n-1) - pk(1));

end

end

figure

plot(t, rate, 'o--');

xlabel('I\_{ext} (uA/cm^2)');

ylabel('Rate of action potentials');

title('Change in neuron firing with current applied');

Q 11

pml=[4, 8.0, 2, 120, -84, -60, 0.0667, -1.2, 18, 12, 17.4, 12, 17.4, 20, 0, 0]';

% figure(1); clf;

options = odeset('OutputFcn', @odeplot, 'Jacobian', @mlodejac, ...

'Vectorized', 'on');

tspan = [0 2000];

iext = [30, -1 ,2000];

setmleparms(pml, iext');

[~, vnull, wnull] = makenulls(1);

vv = @(v) vnull(v) - wnull(v);

appr = [-50, -20, 10];

figure(1); clf;

vlim = [-80 60];

fplot(vnull, vlim);

hold on;

fplot(wnull, vlim);

for i=1:3

e(i,1) = fzero(vv, appr(i));

e(i,2) = vnull(e(i,1));

jac = mlodejac(1,e(i,:));

[tmp1, tmp2] = eig(jac);

plot(e(i,1),e(i,2), 'ko');

eval(i,:) = [tmp2(1), tmp2(4)];

evec{i} = tmp1;

if real(eval(i,1))\*real(eval(i,2)) < 0

ev = evec{i};

y0 = e(i,:) + 0.01\*ev(:,1)';

y1 = e(i,:) + 0.01\*ev(:,2)';

figure

[t,y] = ode15s(@mlode, [0:200], y0, options);

figure

[t,x] = ode15s(@mlode\_reverse, [0:200], y1, options);

figure (1)

plot(y(:,1), y(:,2), x(:,1),x(:,2), 'm')

end

end

xlim([-60 40]); xlabel('Voltage (mV)');

ylim([-0.1 1]); ylabel('w');

title('Phase plane of MLE system');

Q12

pml=[4, 8.0, 2, 120, -84, -60, 0.0667, -1.2, 18, 12, 17.4, 12, 17.4, 20, 0, 0]';

figure(1); clf;

options = odeset('OutputFcn', @odeplot, 'Jacobian', @mlodejac, ...

'Vectorized', 'on');

tspan = [0 2000];

iext = [35, -1 ,2000];

vlim = [-80 60];

for j = 1:1000

iext(1) = iext(1) + 0.005;

setmleparms(pml, iext');

[~, vnull, wnull] = makenulls(1);

vv = @(v) vnull(v) - wnull(v);

e(j,1) = fzero(vv, -20);

e(j,2) = vnull(e(1));

jac = mlodejac(1, e(j,:));

bif = det(jac);

if abs(bif) < 0.00001

bif

iext(1)

fplot(vnull, vlim);

end

if j == 1

fplot(vnull, vlim);

hold on

fplot(wnull, vlim);

end

end

iext = [30; 0; 500];

tspan = [0 500];

for j = 1:150

if j ~= 1

iext(1) = iext(1) + 0.1;

end

setmleparms(pml, iext);

[~, vnull, wnull] = makenulls(1);

vv = @(v) vnull(v)-wnull(v);

ex(1) = fzero(vv,-20);

ex(2) = vv(ex(1));

clf

[t,y] = ode15s(@mlode, tspan, ex);

n = 1;

for k = 1:length(y)-1

if y(k) <= 0 && y(k+1) > 0

pk(n) = t(k+1);

hold on;

plot(t(k+1), y(k+1), 'ko');

n = n+1;

end

end

rate(j) = 0;

if n > 2

rate(j) = (n-2)/(pk(n-1) - pk(1));

end

end

figure

t = 30:0.1:44.9;

plot(t, rate, 'o--');

xlabel('I\_{ext} (uA/cm^2)');

ylabel('Rate of action potentials');

title('Change in neuron firing with current applied');

HHvoltage (Q13 – Q18)

function ydot = hhvoltage(t, y)

%y(1,2,3,4) = v, gk, gna, gl

[params, iext] = gethhparams();

phi = params(8).^((params(7)-6.3)./10);

ydot = zeros(4,1);

if t<=iext(3) && t>= iext(2)

ydot(1) = (iext(1)- params(1).\*y(2).^4.\*(y(1)-params(4)) ...

-params(2).\*y(3).^3.\*y(4).\*(y(1)-params(5)) ...

-params(3)\*(y(1)-params(6)))/params(9) ;

else

ydot(1) = (-params(1).\*y(2).^4.\*(y(1)-params(4)) ...

-params(2).\*y(3).^3.\*y(4).\*(y(1)-params(5)) ...

-params(3)\*(y(1)-params(6)))/params(9) ;

end

ydot(2) = alphan(y(1)).\*(1-y(2)) - betan(y(1)).\*y(2); %ngate

ydot(3) = alpham(y(1)).\*(1-y(3)) - betam(y(1)).\*y(3); %mgate

ydot(4) = alphah(y(1)).\*(1-y(4)) - betah(y(1)).\*y(4); %hgate

function val = alphan(v)

if abs(v+50)>=1.e-4

val = -phi.\*0.01.\*(v+50)./(exp(-(v+50)./10)-1);

else

val = phi.\*0.1./(1-(v+50)./20);

end

end

function val = alpham(v)

if abs(v+50)>=1.e-4

val = -phi.\*0.1.\*(v+35)./(exp(-(v+35)./10)-1);

else

val = phi.\*1./(1-(v+35)./20);

end

end

function val = alphah(v)

val = 0.07.\*phi.\*exp(-(v+60)./20);

end

function val = betan(v)

val = 0.125.\*phi.\*exp(-(v+60)./80);

end

function val = betam(v)

val = 4\*phi\*exp(-(v+60)/18);

end

function val = betah(v)

val = phi./(exp(-(v+30)./10)+1);

end

end

hheqs (Q13 – Q21)

function eqs = hheqs(y)

i =1;

dX = 1;

while ((i < 700) & dX > 0.0001)

i = i+1;

FTY = hhvoltage18(0, y);

[J, b] = numjac(@hhvoltage18, 0, y, FTY, 0.001, [], 0);

dX = -J\FTY;

y = y + dX;

end

dX;

eqs = y;

end

hhsim (Q13 – Q16)

close all

% 1 2 3 4 5 6 7 8 9

% gkbar gnabar glbar ek ena el T Q C

params = [36, 120, 0.3, -72, 55, -50, 6.3, 3, 1];

iext = [-3; 0; 20]; %value, time on, time off

sethhparams(params, iext);

tspan = [0 500];

y0 = [-60.155989; 0.315289; 0.051967; 0.601564];

% dep = 6.6145689701702; %% q 14

% y0(1) = y0(1) + dep;

% options = odeset('OutputFcn', @odeplot);

% Do the simulation

[t, ys] = ode15s(@hhvoltage, tspan, y0);

fprintf('current: v=%f n=%f m=%f h=%f\n', ys(end,:))

% iext = [-3; 0; 20]; %value, time on, time off

% sethhparams(params, iext);

% y2 = ys(end,:)';

% eq = hheqs(y2);

% jac = hhjac(eq);

% es = eig(jac)

% es = eig(jac); %q14, stable

close all

figure;

title('Response of HH model with some sustained external current and rest')

subplot(121);

plot(t, ys(:,1))

xlabel('Time (ms)'); ylabel('Voltage (mV)');

subplot(122);

plot(t, ys(:,2), t, ys(:,3), t, ys(:,4))

xlabel('Time (ms)'); ylabel('Channel activation');

legend('n channel', 'm channel', 'h channel');

%

% figure

% plot(t, ys(:,1))

Q15

close all

% 1 2 3 4 5 6 7 8 9

% gkbar gnabar glbar ek ena el T Q C

params = [36, 120, 0.3, -72, 55, -50.5, 6.3, 3, 1];

iext = [8; -1; 200]; %value, time on, time off

sethhparams(params, iext);

tspan = [0 100];

y2 = [-60.155989; 0.315289; 0.051967; 0.601564];

eq = hheqs(y2);

y = [y2' ; eq'; 0,0,0,0]

% dep = 6.6145689701702; %% q 14

% options = odeset('OutputFcn', @odeplot);

% Do the simulation

for i = 1:3

[t, ys] = ode15s(@hhvoltage, tspan, y(i,:)');

plot (ys(:,1), ys(:,2));

hold on

end

% close all

% figure;

% title('Response of HH model with some sustained external current and rest')

% subplot(121);

% plot(t, ys(:,1))

% xlabel('Time (ms)'); ylabel('Voltage (mV)');

% subplot(122);

% plot(t, ys(:,2), t, ys(:,3), t, ys(:,4))

% xlabel('Time (ms)'); ylabel('Channel activation');

% legend('n channel', 'm channel', 'h channel');

%

%

% figure

% plot(t, ys(:,1))

Q17

close all

% 1 2 3 4 5 6 7 8 9

% gkbar gnabar glbar ek ena el T Q C

params = [36, 120, 0.3, -72, 55, -50, 6.3, 3, 1];

iext = [-3; 0; 20]; %value, time on, time off

sethhparams(params, iext);

tspan = [0 200];

y0 = [-60; 0.315289; 0.051967; 0.601564];

y1 = [-63.5493; 0.2641; 0.0346; 0.7081];

% dep = 6.6145689701702; %% q 14

% y0(1) = y0(1) + dep;

% options = odeset('OutputFcn', @odeplot);

% Do the simulation

[t, ys] = ode15s(@hhvoltage17, tspan, y0, options);

[t, yd] = ode15s(@hhvoltage17, tspan, y1, options);

y2 = ys(end,:)';

eq = hheqs(y2);

jac = hhjac(eq);

es = eig(jac)

es = eig(jac); %q14, stable

figure

plot(ys(:,1), ys(:,3))

hold on

plot(yd(:,1), yd(:,3), 'k')

xlabel('Voltage (mV)');

ylabel('m value');

title('Phase plane of anode break spike');

legend('No hyperpolarization', 'Hyperpolarization current applied');

HHvoltage18 (Q18)

function ydot = hhvoltage18(t, y)

%y(1,2,3,4) = v, gk, gna, gl

[params, iext] = gethhparams();

phi = params(8).^((params(7)-6.3)./10);

phi =1;

ydot = zeros(2,1);

a = 0.95; b =-1.05;

if t<=iext(3) && t>= iext(2)

ydot(1) = (iext(1)- params(1).\*y(2).^4.\*(y(1)-params(4)) ...

-params(2).\*(alpham(y(1))/(alpham(y(1))+betam(y(1)))).^3.\*(a + b\*y(2)).\*(y(1)-params(5)) ...

-params(3)\*(y(1)-params(6)))/params(9) ;

else

ydot(1) = (-params(1).\*y(2).^4.\*(y(1)-params(4)) ...

-params(2).\*minfhh(y(1)).^3.\*(a + b\*y(2)).\*(y(1)-params(5)) ...

-params(3)\*(y(1)-params(6)))/params(9) ;

end

ydot(2) = alphan(y(1))\*(1-y(2)) - betan(y(1))\*y(2); %ngate

function val = alphan(v)

if abs(v+50)>=1.e-4

val = -phi.\*0.01.\*(v+50)./(exp(-(v+50)./10)-1);

else

val = phi.\*0.1./(1-(v+50)./20);

end

end

function val = alpham(v)

if abs(v+50)>=1.e-4

val = -phi.\*0.1.\*(v+35)./(exp(-(v+35)./10)-1);

else

val = phi.\*1./(1-(v+35)./20);

end

end

function val = betan(v)

val = 0.125.\*phi.\*exp(-(v+60)./80);

end

function val = betam(v)

val = 4\*phi\*exp(-(v+60)/18);

end

end

Sim18 (Q 18 – Q 21)

close all

% 1 2 3 4 5 6 7 8 9

% gkbar gnabar glbar ek ena el T Q C

params = [36, 120, 0.3, -72, 55, -49.3, 6.3, 3, 1];

iext = [-5; 30; 60]; %value, time on, time off

sethhparams(params, iext);

tspan = [0 100];

y0 = [-60.155989; 0];

% options = odeset('OutputFcn', @odeplot);

% Do the simulation

[t, ys] = ode15s(@hhvoltage18, tspan, y0);

close all

figure;

title('Response of HH model with some sustained external current and rest')

% subplot(121);

plot(t, ys(:,1))

xlabel('Time (ms)'); ylabel('Voltage (mV)');

% subplot(122);

% plot(t, ys(:,2))

% xlabel('Time (ms)'); ylabel('Channel activation');

% legend('n channel');

%

% figure

% plot(t, ys(:,1))

Q20

params = [36, 120, 0.3, -72, 55, -49.3, 6.3, 3, 1];

iext = [-40; 0; 20]; %value, time on, time off

sethhparams(params, iext);

figure(1); clf;

vlim = [-80 60];

y0 = [-63 , 0.27; -60, 0.32];

for i=1:2

[t,y] = ode15s(@hhvoltage, [0:400], y0(i,:));

figure (1)

plot(y(:,1), y(:,2))

hold on

pause

end

xlim([-60 40]); xlabel('Voltage (mV)');

ylim([-0.1 1]); ylabel('w');

title('Phase plane of HH system');