Weekly Homework 3

Benjamin Cramer, Julian Göltz
Brain Inspired Computing

November 2, 2015

Exercise 3.1. Stability Conditions in 2D

a) Requirement for stability $r_\pm < 0$ with $\lambda_\pm = r \pm i\omega$; eigenvalues given by:

$$\lambda_\pm = \frac{1}{2}\left(F_u + G_w \pm \sqrt{(F_u + G_w)^2 - 4(F_u G_w - F_w G_u)}\right) \tag{1}$$

if $\sqrt{\cdots}$ gets imaginary $\Rightarrow r_\pm < 0$ since $F_u + G_w < 0$ (Condition 1) if $\sqrt{\cdots}$ gets real:

$$\sqrt{(F_u + G_w)^2 - 4(F_u G_w - F_w G_u)} < \sqrt{(F_u + G_w)^2} = |F_u + G_w| \tag{2}$$

where we used condition number two; plugging this into eigenvalue expression leads us to:

$$\lambda_\pm \le \frac{1}{2}(F_u + G_w \pm |F_u + G_w|) \le 0 \tag{3}$$

in both cases the system is stable since $r_\pm < 0$

b) Using the following Mathematica code to obtain fixed points and decide whether they are stable:

```
eps = 0.1;
a = 15 / 8;
b = 3 / 2;
II = 0;
F[u_, w_] = u - u^3 / 3 - w + II;
G[u_, w_] = eps (a + b u - w);
sol = Solve[F[u, w] == 0 && G[u, w] == 0, {u, w}]
Fu = D[F[u, w], u] /. sol;
Fw = D[F[u, w], w] /. sol;
Gu = D[G[u, w], u] /. sol;
Gw = D[G[u, w], w] /. sol;
(Fu + Gw) < 0
(Fu Gw - Fw Gu) > 0
```

Solve::ratnz : Solve was unable to solve the system with inexact coefficients. The answer was obtained by solving a corresponding exact system and numericizing the result. ≫

```
{{u → -1.5, w → -0.375}, {u → 0.75 - 1.78536 i, w → 3. - 2.67804 i},
 {u → 0.75 + 1.78536 i, w → 3. + 2.67804 i}}

{-1.35, 3.525 + 2.67804 i, 3.525 - 2.67804 i} < 0

{0.275, -0.2125 - 0.267804 i, -0.2125 + 0.267804 i} > 0
```

```
eps = 0.1;
a = 15 / 8;
b = 3 / 2;
II = 15 / 8;
F[u_, w_] = u - u^3 / 3 - w + II;
G[u_, w_] = eps (a + b u - w);
sol = Solve[F[u, w] == 0 && G[u, w] == 0, {u, w}]
Fu = D[F[u, w], u] /. sol;
Fw = D[F[u, w], w] /. sol;
Gu = D[G[u, w], u] /. sol;
Gw = D[G[u, w], w] /. sol;
(Fu + Gw) < 0
(Fu Gw - Fw Gu) > 0
```

Solve::ratnz : Solve was unable to solve the system with inexact coefficients. The answer was obtained by solving a corresponding exact system and numericizing the result. ≫

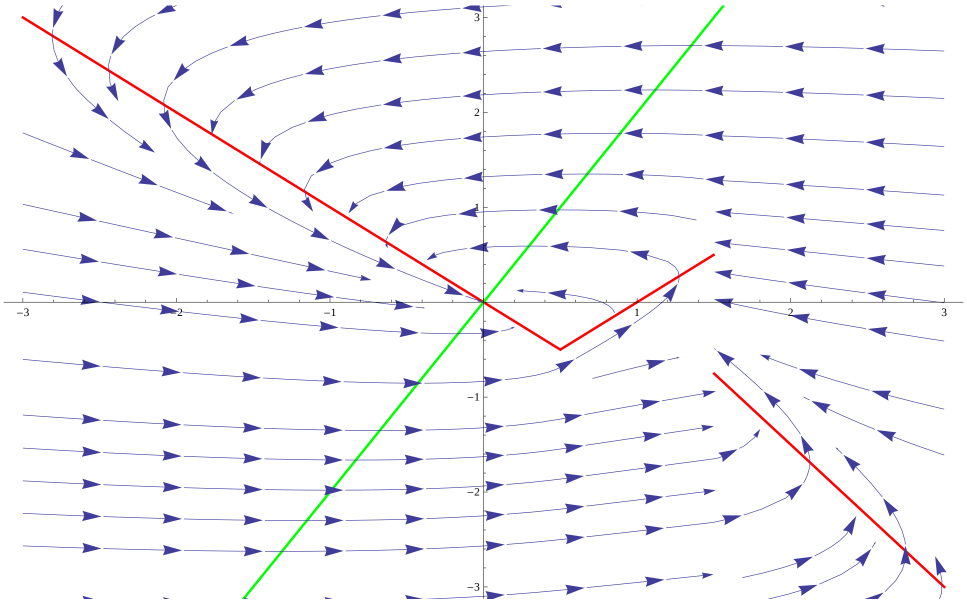
```
{{u → 0., w → 1.875}, {u → 0. - 1.22474 i, w → 1.875 - 1.83712 i},
 {u → 0. + 1.22474 i, w → 1.875 + 1.83712 i}}

{0.9, 2.4 + 0. i, 2.4 + 0. i} < 0

{0.05, -0.1 + 0. i, -0.1 + 0. i} > 0
```

From this we follow that for $I = 0$ there exists a stable FP (the first result: at $u = -1.5$ and $w = -0.375$, these values are used as initial values below). For the other case there is no FP.

Exercise 3.2. Piecewise linear nullclines

a) nullclines and flow in Phase plane of FitzHugh-Nagumo model using Mathematica and an input current of $I = 0$:

```
II = -2;

a = -1;
c1 = -1;
b = 2;
eps = 0.1;
c0 [u_] = -0.5 u - 1.5 c1;
NCw[u_] = b u;


NCu[u_] = II + a u HeavisideTheta[-(u - 0.5)] + a (1 - u) HeavisideTheta[u - 0.5]
    HeavisideTheta[-(u - 1.5)] + (c0[u] + c1 u ) HeavisideTheta[u - 1.5];
nullcline1 = Plot[NCu[u], {u, -3, 3}, PlotStyle → {Red, Thick}];
nullcline2 = Plot[NCw[u], {u, -3, 3}, PlotStyle → {Green, Thick}];

Du[u_, w_] = NCu[u] - w;(* II already in there*)
Dw[u_, w_] = eps ( b u - w);
flow = StreamPlot[{Du[u, w], Dw[u, w]}, {u, -3, 3}, {w, -6, 6}];
(*SetDirectory[ ]
Export["excercise_3.2_I0.jpg",*)
 Show[nullcline1, nullcline2, flow]
```



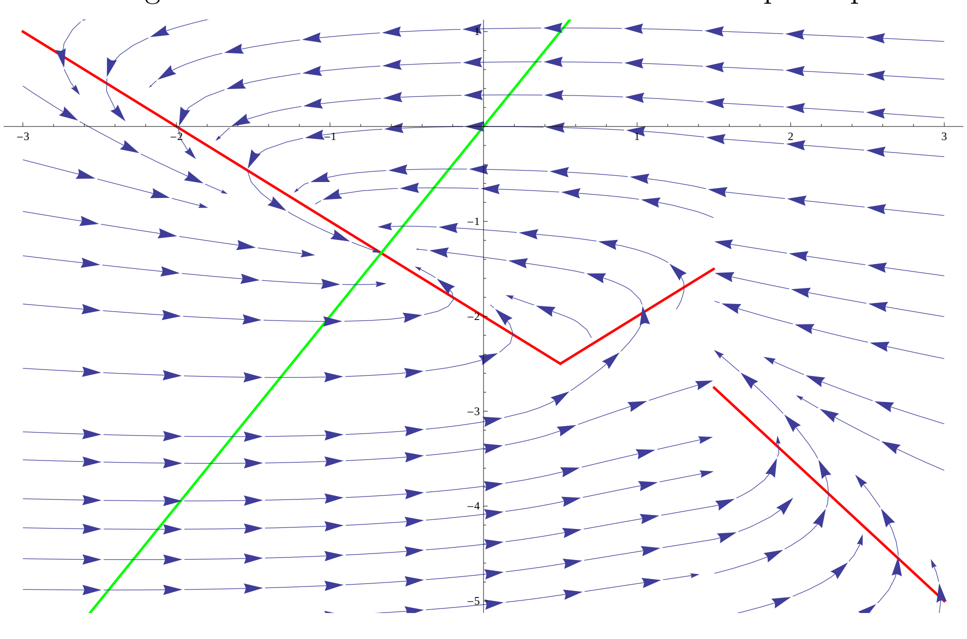
b) Same as in a but with an input current of $I = -2$. The fixed point is calculated by the intersection of the following equations ($u < 0.5$ compare figure):

$$0 = au - w - 2 \tag{4}$$
$$0 = bu - w \tag{5}$$

which leads us to:

$$u = \frac{1}{a-b} = -\frac{2}{3} \tag{6}$$
$$w = au - 2 = -\frac{4}{3} \tag{7}$$

the next figure shows the nullclines and the flow in the phase plane for $I = -2$

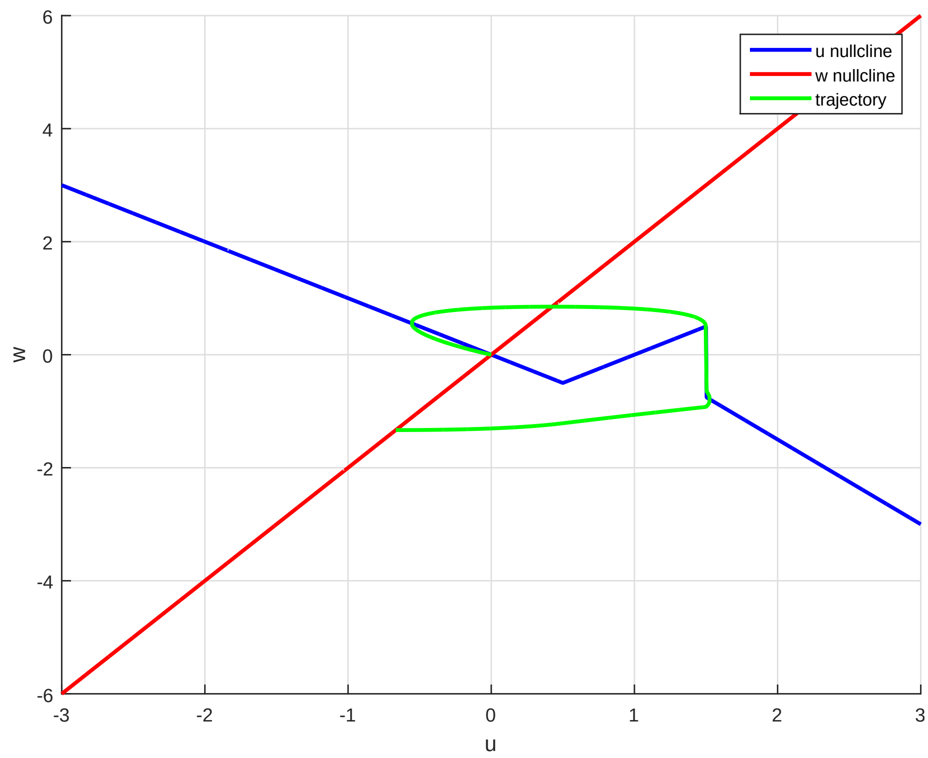


c) MATLAB code for simulation of the nullclines and trajectory in phase plane:

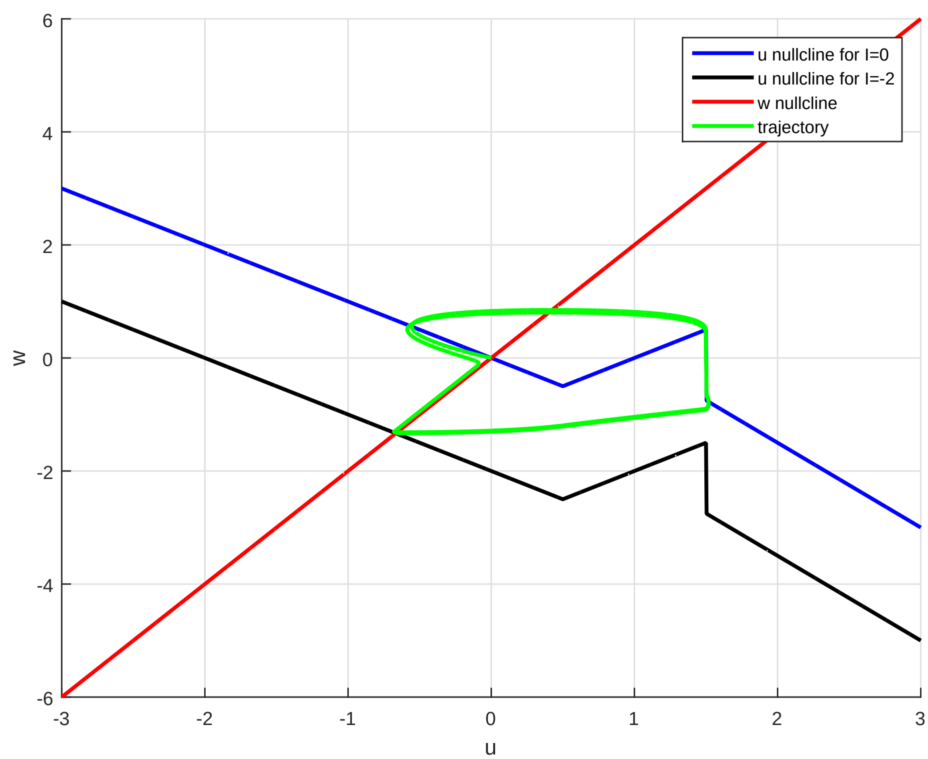
```
close all
clear all
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 0. PARAMETERS

% Euler integration
h = 10e-5;      % step size parameter
time = 500;     % simulation time

% Model
a = -1;
c1 = -1;
b = 2;
epsilon = 0.1;

t = 0:h:time; % generate time vector
I = [-1*(linspace(0,2,round(length(t)/2-1))) zeros(round(length(t)/2),1)']'; % create current vector

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% I. CALCULATE NULLCLINES

u = linspace(-3,3,length(t)); % preallocate voltage array
w = linspace(-3,3,length(t)); % preallocate w array
w = zeros(length(u),1); % preallocate aray for piecewise function

for i = 1:length(u)
    u1 = u(i);
    w(i) = fu(u1,a,c1);
end

nu = w + 0; % calculate voltage nullcline
nu_new = w - 2; % calculate voltage nullcline
nw = b*u; % calculate w nullcline
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% II. PERFORM INTEGRATION

u_sol = zeros(size(t)); % Preallocate array for velocities
w_sol = zeros(size(t)); % Preallocate array for positions

u_sol(1) = -2/3;%1.5;               % Initial condition gives solution for position at t=0.
w_sol(1) = -4/3;%0.375;             % Initial condition gives solution for velocity at t=0.

for i=1:(length(t)-1) % loop over time
    u_sol(i+1) = u_sol(i) + (fu(u_sol(i),a,c1) - w_sol(i) + I(i))*h; % integrate voltage DE
    w_sol(i+1) = w_sol(i) + epsilon*(b*u_sol(i) - w_sol(i))*h; % integrate w DE
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% III. PLOT REULTS
figure
hold on
grid on
plot(u,nu,'b','linewidth',2)
plot(u,nu_new,'k','linewidth',2)
plot(u,nw,'r','linewidth',2)
plot(u_sol,w_sol,'g','linewidth',2)
legend('u nullcline for I=0','u nullcline for I=-2','w nullcline','trajectory')
xlabel('u')
ylabel('w')
print(gcf,'-depsc','exercise32c_full.eps')
```

```
function f = fu(u,a,c1)
c0 = -0.5*u -1.5*c1;
if u<0.5
    f = a*u;
elseif 0.5 <= u && u < 1.5
    f = a*(1-u);
elseif u >= 1.5
    f = c0 + c1*u;
end
end
```

The next figure shows the trajectory in phase plane after the switch-off of the hyperpolarization current. The initial values for the Euler integration correspond to the calculated fixed point (Equation 7)



The next figure shows the full trajectory in phase plane for a linearly increasing hyperpolarizing current, which is suddenly switched of. The initial values for the Euler integration are chosen to be $w = u = -2$. The w nullcline is shown for $I = 0$ and $I = -2$

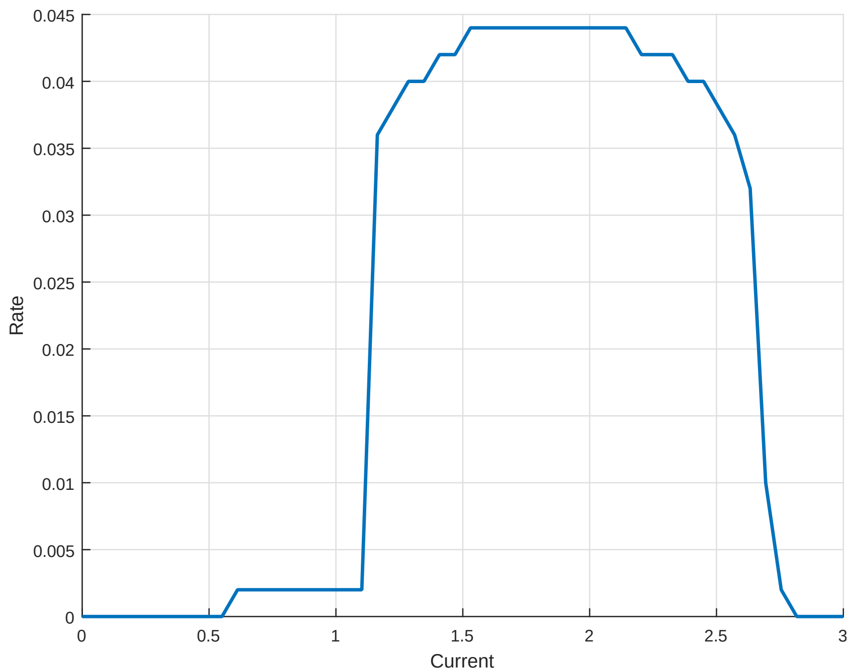


Exercise 3.3. Exploring the FitzHugh model

a) MATLAB code for Euler integration of FitzHugh-Nagumo model:

```
close all
clear all
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% I. DEFINE PARAMETERs

% Euler
h = 10e-5; % step size parameter
time = 500; % simulation time
t = 0:h:time; % generate time vector

% model
epsilon = 0.1;
a = 15/8;
b = 3/2;
I = linspace(0,3,50); % create linear incresing current vector

rate = zeros(50,1); % preallocate rate array

for j = 1:length(I) % loop over currents
curr = I(j);
u = zeros(size(t)); % Preallocate array for velocities
w = zeros(size(t)); % Preallocate array for positions

u(1) = -1.5; % Initial condition gives solution for position at t=0.
w(1) = -0.375; % Initial condition gives solution for velocity at t=0.

numberOfPeaks = 0; % set counter
alreadyPeaked = 0; % set counter
threshold = 1; % set threshold

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% II. PERFORM INTEGRATION
for i=1:(length(t)-1)
    u(i+1) = u(i) + (u(i) -u(i)^3/3 -w(i) + curr)*h; % integrate u
    w(i+1) = w(i) + epsilon*(a + b*u(i) - w(i))*h; % inegrate w
    % detection algorithm
    if(u(i+1) >= threshold)
        alreadyPeaked = 1;
    else
        if(alreadyPeaked == 1)
            alreadyPeaked = 0;
            numberOfPeaks = numberOfPeaks + 1;
        end
    end
end
rate(j) = numberOfPeaks/time; % normalize rate
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% III. PLOT RESULTS

figure                      % prepare figure
hold on                     % plot in every loop cycle in same figure
grid on                     % plot mesh grid
xlabel('Current')
ylabel('Rate')
plot(I,rate,'Linewidth',2)
print(gcf,'-depsc','ecxercise3ac.eps')
```

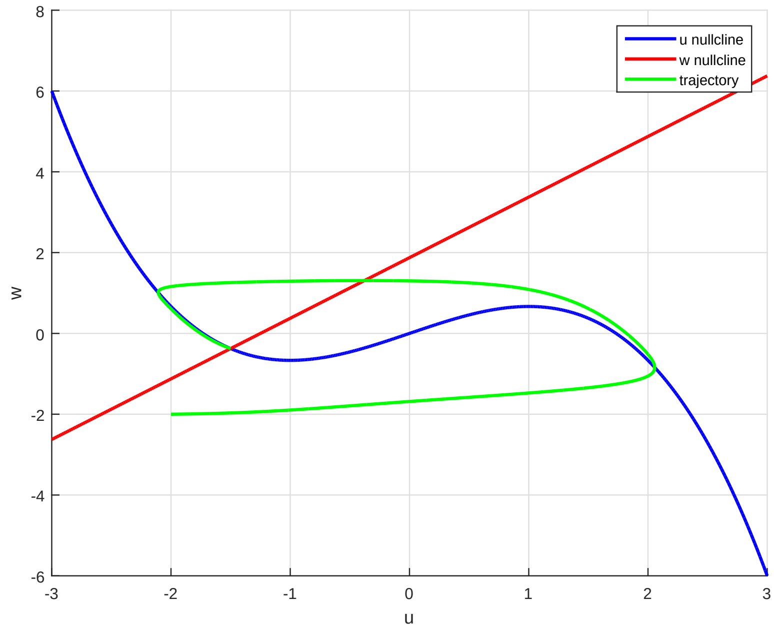
The next plot shows the firing rate as a function of input current



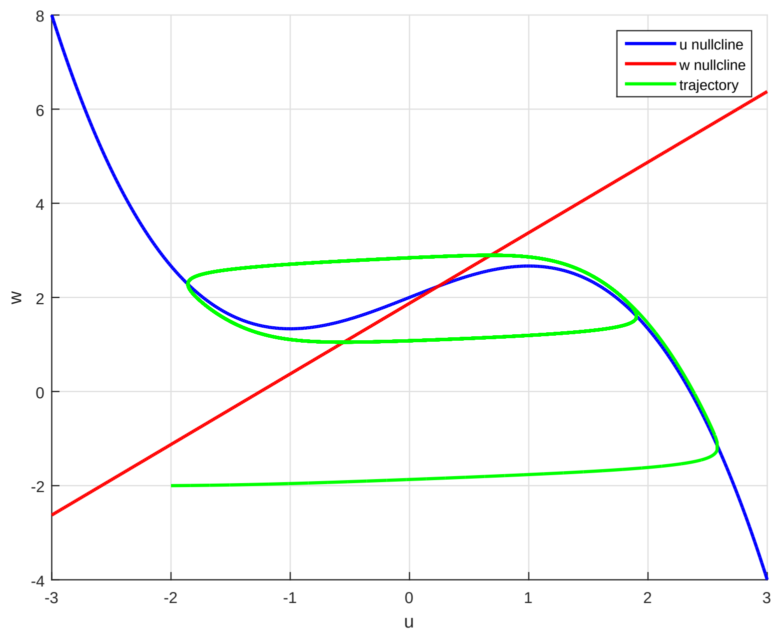
b) MATLAB code for calculating nullclines and an example trajectory

```
close all
clear all
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% I. DEFINE PARAMETERS

% Euler integration
h = 10e-5; % step size parameter
time = 500; % simulation time
t = 0:h:time; % generate time vector

% model
a = 15/8;
b = 3/2;
epsilon = 0.1;
I = 2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% II. CALCULATE NULLCLINES

u0 = linspace(-3,3,20000);

w1 = u0 - u0.^3/3 + I; % calculate u nullcline
w2 = a + b*u0; % calculate w nullcline

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% III. SOLVE DEs

u = zeros(size(t)); % Preallocate array for velocities
w = zeros(size(t)); % Preallocate array for positions

u(1) = -2; % Initial condition gives solution for position at t=0.
w(1) = -2; % Initial condition gives solution for velocity at t=0.

for i=1:(length(t)-1)
    u(i+1) = u(i) + (u(i) -u(i)^3/3 -w(i) + I)*h;
    w(i+1) = w(i) + epsilon*(a + b*u(i) - w(i))*h;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%
%% IIII. PLOT RESULTS

figure
hold on
grid on
plot(u0,w1,'b','linewidth',2)
plot(u0,w2,'r','linewidth',2)
plot(u,w,'g','linewidth',2)
legend('u nullcline','w nullcline','trajectory')
xlabel('u')
ylabel('w')
print(gcf,'excercise33c_I2.pdf')
```

Plot of nullclines and trajectory at $I = 0$ where $\nu = 0$



Plot of nullclines and trajectory at $I = 2$ where $\nu \neq 0$



We interpret the plot as follows: in the first case the fixpoint in the phase space is reached fast, and the coordinates stay do not leave again. This means that the neuron adapts to the new state, i.e. the ion channels are opened until the equlibrium is reached. The current to which the cell was exposed is not large enough to induce spiking.
On the contrary, for the second case no fixpoint is reached, the coordinates keep moving in the phase space. This behavior corresponds to a spiking trail, i.e. the used current was large enough for spiking. This is in agreement with the plot of the spiking frequency vs. the input current.