Neuronal Dynamics: Python Exercises

Professor Wulfram Gerstner
Laboratory of Computational Neuroscience, EPF Lausanne

Phase Plane and Bifurcation Analysis

Exercise 1 Python phase plane analysis

Download `fitznagumo.py` from book's webpage. Create a script file called `answers.py` and add the following header:

```
from fitznagumo import *
```

You will type the code for your answers right below, adding more code with each exercise. When you want to execute your code, open ipython in a terminal and type:

```
>> import answers
```

In this exercise you will need to plot data, which can be done using pylab.

1. Use the function `plot` to plot the two nullclines of the Fitzhugh-Nagumo system given in Eq. 1 for $I = 0$ and $\varepsilon = 0.1$. Plot them in the $u - w$ plane, for voltages in the region $u \in [-2.5, 2.5]$. For instance the following example shows plotting the function $y(x) = -\frac{x^2}{2} + x + 1$:

   ```
   x = arange(-2.5,2.51,.1) # create an array of x values
   y = -x**2/2. + x + 1 # calculate the function values for the given x values
   plot(x,y,color='black') # plot y as a function of x
   xlim(-2.5,2.5) # constrain the x limits of the plot
   ```

   You can use similar code to plot the nullclines.

2. Get the two lists `u,w` by calling `u,w = get_trajectory`(u_0, w_0, I) for $u_0 = 0$, $w_0 = 0$ and $I = 1.3$. They are corresponding values of u and w during trajectories starting at the given point (u_0, w_0) for a given constant current I. Plot the nullclines for this given current and the trajectories into the $u - w$ plane.

3. At this point for the same current I, call the function `plot_flow`(I), which adds the flow created by the system Eq. 1 to your plot. This indicates the direction that trajectories will take. If everything went right so far, the trajectories should follow the flow.

4. First, create a new figure by calling `figure()` and then plot the u data points from the trajectory obtained in exercise 2 on the ordinate. You can do this by using the plot function and passing only the array of u data points:

   ```
   u = [1,2,3,4] # example data points of the u trajectory
   plot(u, color='blue') # plot will assume that u is the ordinate data
   ```

5. Finally, change the input current in your python file to other values $I > 0$ and reload it. You might have to first define I as a variable and then use this variable in all following commands if you did not do so already. At which value of I do you observe the change in stability of the system?

Exercise 2 Graphical bifurcation analysis

Consider the following two-dimensional Fitzhugh-Nagumo model:

$$\left[\begin{array}{rcl} \dfrac{du}{dt} & = & u\left(1-u^2\right) - w + I \equiv F(u,w) \\ \dfrac{dw}{dt} & = & \varepsilon\left(u - 0.5w + 1\right) \equiv \varepsilon G(u,w)\,, \end{array}\right. \tag{1}$$

The linear stability of a system of differential equations can be evaluated by calculating the eigenvalues of the system's Jacobian at the fixed points. In the following we will graphically explore the linear stability of the fixed point of the system Eq. 1. We will find that the linear stability changes as the input current crosses a critical value.

Set $\varepsilon = .1$. Create the variable I and set it to zero for the moment.

1. *(Jacobian function & eigenvalues)* The Jacobian of Eq. 1 as a function of the fixed point is given by

$$J\left(u_0, w_0\right) = \begin{pmatrix} 1 - 3u_0^2 & -1 \\ 0.1 & -0.05 \end{pmatrix}$$

   a) Write a python function `get_jacobian`(u_0, w_0) that returns the Jacobian evaluated for a given fixed point (u_0, v_0) as a python list. *Hint*: An example for a function that returns a list corresponding to the matrix $M(a,b) = \begin{pmatrix} a & 1 \\ 0 & b \end{pmatrix}$ is:

```
def get_M(a,b):
        return [[a,1],[0,b]] # return the matrix
```

   b) Executing `u0,w0 = get_fixed_point`(I) gives you the numerical coordinates of the fixed point for a given current I. Use the function you created in 1 to evaluate the Jacobian at this fixed point and store it in a new variable `J`.

   c) Calculate the eigenvalues of the Jacobian `J`, which you computed in b, by using the function `linalg.eigvals(J)`. Both should be negative for $I = 0$.

2. *(Bifurcation analysis)* Wrap the code you wrote so far by a loop, to calculate the eigenvalues for increasing values of I. Store the changing values of each eigenvalue in seperate lists, and finally plot their real values against I. Use this example to help you getting started:

```
list1 = []
list2 = []
currents = arange(0,4,.1) # the I values to use
for I in currents:
        # your code to calculate the eigenvalues e = [e1,e2] for a given I goes here
        list1.append(e[0].real) # store each value in a separate list
        list2.append(e[1].real)
# plot list1 and list 2
```

In what range of I are the real parts of eigenvalues positive? Compare this with your result from 1.5. What does this imply for the stability of the fixed point? What has become stable in this system instead of the fixed point?