Brain Inspired Computing - Problem Set 1

Sven Bordukat, Paul Meehan

May 15, 2019

Exercise 1

a) Let $A := F_u + G_w$, $B := F_u G_w - F_w G_u$ and $C := (F_u + G_w)^2 - 4(F_u G_w - F_w G_u)$. Assume $A > 0$. From the eigenvalue equation

$$\begin{aligned}\lambda_\pm &= \frac{1}{2}(F_u + G_w \pm \sqrt{(F_u + G_w)^2 - 4(F_u G_w - F_w G_u)}) \\ &= \frac{1}{2}(A \pm \sqrt{A^2 - 4B})\end{aligned}$$

we derive that if $C \geq 0$, $\lambda_+ > 0$ as well. If $C < 0$, $\lambda_\pm$ can be expressed through $\lambda_\pm = r \pm i * \omega$ with $r = A/2 > 0$. However, any positive λ leads to growth, which means that the model is not stable under these conditions. In the case of $C > 0$, we have a saddle or an unstable situation, depending on whether λ_- is positive or negative; in the case of $C < 0$ we have a growing spiral. Therefore, $A < 0$ must hold true. In the case of $C < 0$, this is enough, as any solution that satisfies $A < 0$ will lead to $\lambda_\pm = r \pm i * \omega$ with $r < 0$. However, for $C > 0$, we need to make sure that $\lambda_+ < 0$ in any case. To satisfy this condition, $\sqrt{C} < abs(A) = \sqrt{A^2}$ must hold true. Expansion of C leads to $\sqrt{A^2 - 4B} < \sqrt{A^2}$. From this we can easily see that $F_u G_w - F_w G_u = B > 0$ must hold true.

b) The fixpoints are $(-\frac{3}{2}, -\frac{3}{8})$ and $(0, \frac{15}{8})$, respectively. For the case of $I = 0$ we can simply calculate the conditions given above. We have

$$\begin{aligned}F_u &= 1 - u^2 \\ F_w &= -1 \\ G_u &= \epsilon * b \\ G_w &= -\epsilon * w\end{aligned}$$

which gives us

$$\begin{aligned}F_u + G_w &= 1 - u^2 - \epsilon * w \\ &= 1 - \frac{9}{4} + 0.1 * \frac{3}{8} \\ &= \frac{8 - 18 + 0.3}{8} \\ &= -\frac{9.7}{8} < 0\end{aligned}$$

and

$$\begin{aligned}F_u * G_w - F_w * G_u &= (1 - u^2) * (-\epsilon * w) - (-1 * \epsilon * b) \\ &= \epsilon(-\frac{5}{8} * \frac{3}{8} + \frac{3}{2}) \\ &= \epsilon(\frac{3}{2} - \frac{15}{64}) > 0\end{aligned}$$

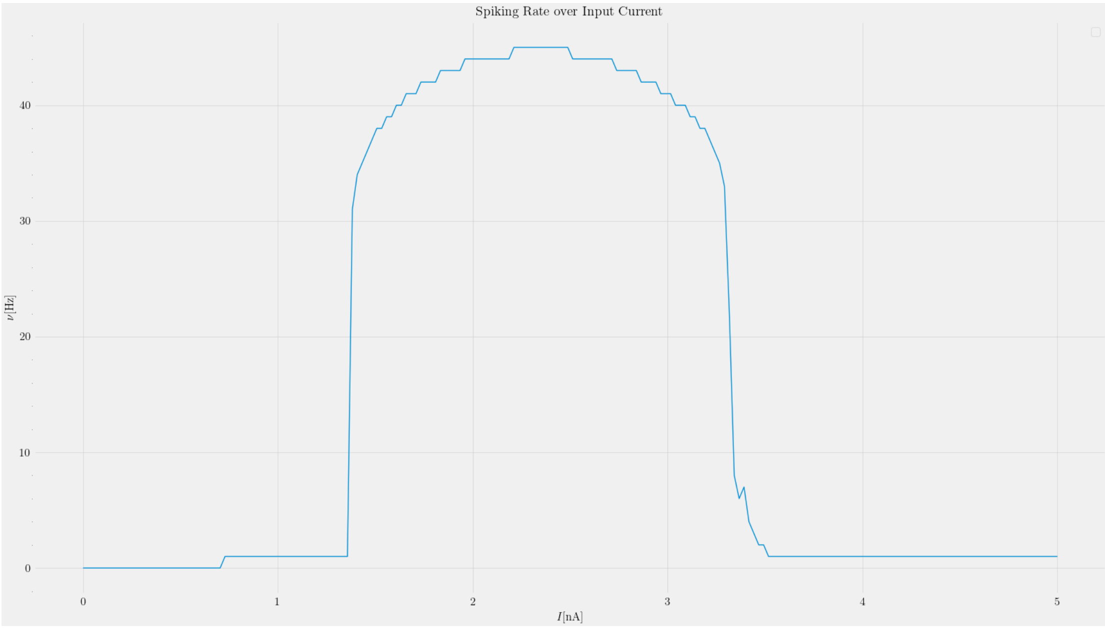
Therefore the fixpoint for $I = 0$ is stable.
The nullcline of w is given through $G = 0 \Leftrightarrow w = a + b * u$. For $I = 15/8$ lets look at a point on the nullcline of w at coordinates $(u_0 + \delta u, a + b * (u_0 + \delta u))$ with $\delta u > 0, \delta u \ll 1$. As $u_0 = 0$, we get

$$\begin{aligned} F &= \delta u - \frac{\delta u^3}{3} - \frac{15}{8} + \delta u * \frac{3}{2} + \frac{15}{8} \\ &\approx \frac{5}{2}\delta u > 0 \end{aligned}$$

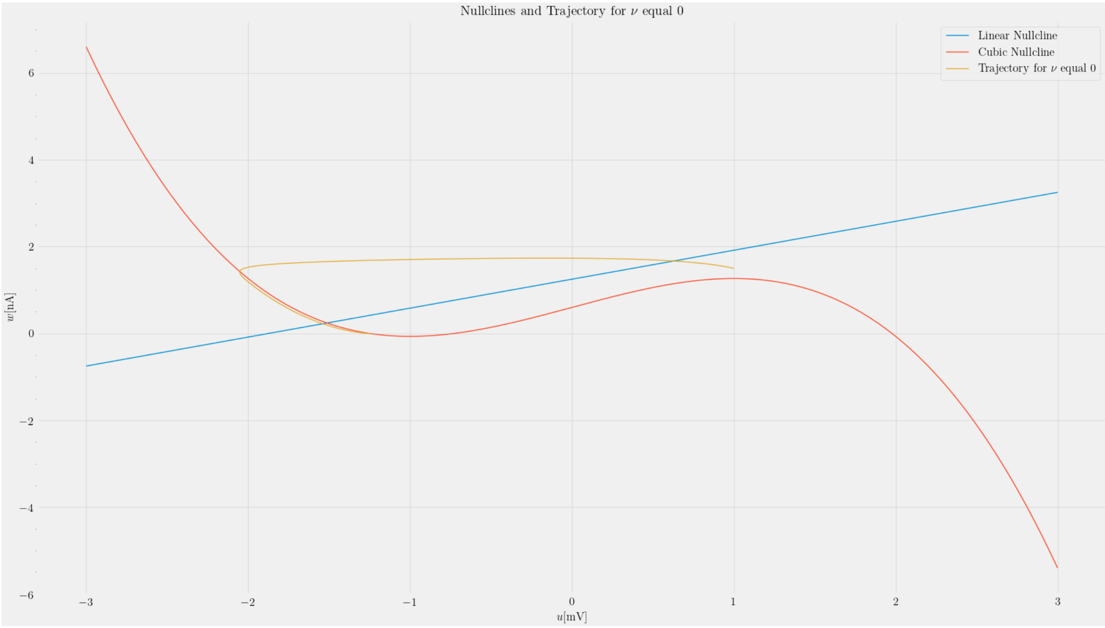
This means that the arrows point away from the fixpoint (at least on the nullcline) and the fixpoint is therefore not stable.

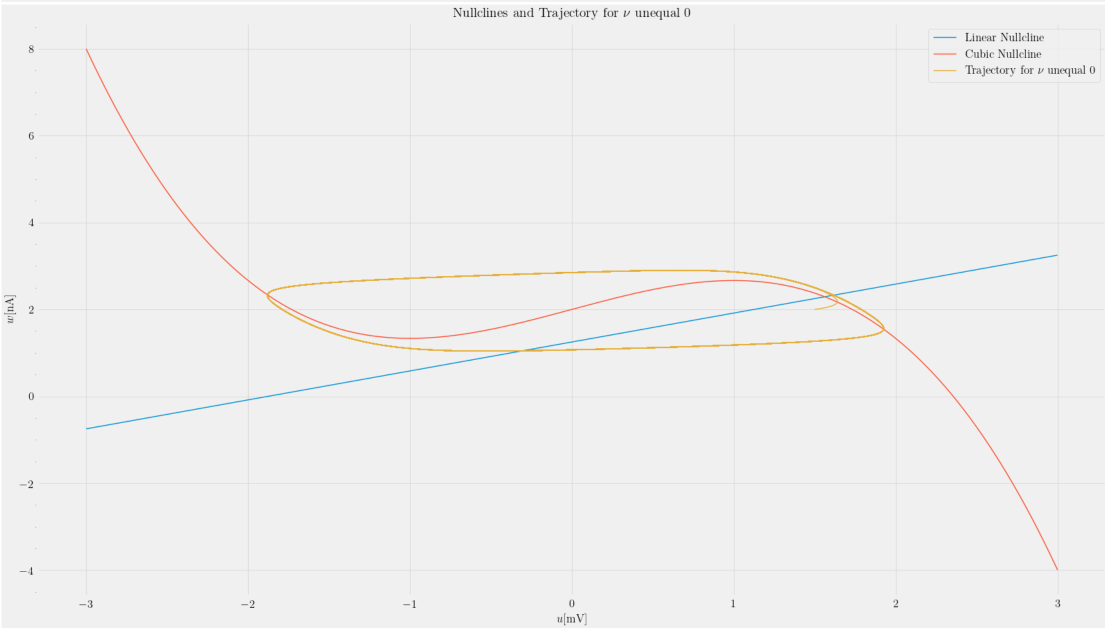
Exercise 3

a)



b)


Nullclines and Trajectory for ν equal 0
Linear Nullcline
Cubic Nullcline
Trajectory for ν equal 0
u[mV]
w[nA]


Nullclines and Trajectory for ν unequal 0
Linear Nullcline
Cubic Nullcline
Trajectory for ν unequal 0
u[mV]
w[nA]

Code

```python
#!/usr/bin/env python3
# Set-up PGF as the backend for saving a PDF
import matplotlib
from matplotlib.backends.backend_pgf import FigureCanvasPgf
matplotlib.backend_bases.register_backend('pdf', FigureCanvasPgf)
import textwrap as tw
from math import floor, log10
import numpy as np
import matplotlib.pyplot as plt
import copy
from scipy.integrate import odeint as integrate

#
# Settings and functions for the plots.
#
plt.style.use('fivethirtyeight')

pgf_with_latex = {
    "pgf.texsystem": "xelatex",          # Use xetex for processing
    "text.usetex": True,                 # use LaTeX to write all text
    "font.family": "serif",              # use serif rather than sans-serif
    "font.serif": "Linux Libertine",     # use Libertine as the font
    "font.sans-serif": "Linux Biolinum",# use Biolinum as the sans-serif font
    "axes.labelsize": 12,
    "font.size": 12,
    "legend.fontsize": 12,
    "axes.titlesize": 14,            # Title size when one figure
    "xtick.labelsize": 12,
    "ytick.labelsize": 12,
    "figure.titlesize": 14,          # Overall figure title
    "pgf.rcfonts": False,            # Ignore Matplotlibrc
    "pgf.preamble": [                # Set-up LaTeX
        r'\usepackage{fontspec}',
        r'\setmainfont{Linux Libertine}',
        r'\usepackage{unicode-math}',
        r'\setmathfont{Linux Libertine}'
    ]
}

matplotlib.rcParams['grid.color'] = '#cccccc'
matplotlib.rcParams['grid.linestyle'] = '-'
matplotlib.rcParams['grid.linewidth'] = 0.4
matplotlib.rcParams['ytick.minor.visible'] = 'True'
matplotlib.rcParams['ytick.minor.right'] = 'True'
matplotlib.rcParams['ytick.minor.size'] = '2.0'
matplotlib.rcParams['ytick.minor.width'] = '0.6'
matplotlib.rcParams.update(pgf_with_latex)

# Define function for string formatting of scientific notation
def sci_notation(num, decimal_digits=1, precision=None, exponent=None):
    """
```

```
    Returns a string representation of the scientific
    notation of the given number formatted for use with
    LaTeX or Mathtext, with specified number of significant
    decimal digits and precision (number of decimal digits
    to show). The exponent to be used can also be specified
    explicitly.
    """
    if num == 0:
        return '0'
    if not exponent:
        exponent = int(floor(log10(abs(num))))
    coeff = round(num / float(10**exponent), decimal_digits)
    if not precision:
        precision = decimal_digits
    if coeff - 1 < 10**(-decimal_digits):
        return r"$10^{{{0:d}}}$".format(exponent)

    return r"${0:.{2}f}\cdot10^{{{1:d}}}$".format(coeff, exponent, precision)

# Customize the given axis.
def cplot(axis, xlabel=None, ylabel=None, xscale='linear', yscale='linear',
          title=None, grid=False, tick_style=None, legend=True,
          xlim=None, ylim=None):
    axis.set_xlabel(xlabel)
    axis.set_ylabel(ylabel)
    axis.set_xscale(xscale)
    axis.set_yscale(yscale)
    axis.set_xlim(xlim)
    axis.set_ylim(ylim)
    axis.set_title(title)
    if tick_style=='percent':
        vals = axis.get_yticks()
        axis.set_yticklabels(['{:,.2%}'.format(x) for x in vals])
    if tick_style=='sci_not':
        vals = axis.get_yticks()
        axis.set_yticklabels([sci_notation(x) for x in vals])
    axis.grid(grid)
    if legend:
        axis.legend()

#
# Euler algorithm from previous exercise.
#
def euler_step(f, x, t, step_size):
    dx = step_size * f(x,t)
    x += dx
    t += step_size
    return x, t

def euler(f, step_size, t_sim, x_0=0, t_0=0):
    x, t = [x_0], [t_0]
    while t[-1] < t_sim:
        x_t, t_t = euler_step(f, copy.deepcopy(x[-1]), copy.deepcopy(t[-1]),
                              step_size)
```

```
            x.append(x_t)
            t.append(t_t)
    return x, t

#
# FitzHugh-Nagumo
#
def fitzHugh_Nagumo(x, t):
    epsilon, a, b = 0.1, 15.0/8.0, 3.0/2.0
    u = x[0] - x[0]**3 / 3 - x[1] + x[2]
    w = epsilon * (a + b * x[0] - x[1])
    return np.array([u,w,0])

def spike_count(x):
    counts, thresh = 0, 1.0
    for i, y in enumerate(x):
        counts += int(y<thresh and x[(i+1)%len(x)]>thresh)
    return counts

def lin_nullcline(u):
    a, b = 15.0/8.0, 3.0/2.0
    return (u + a)/b

def cub_nullcline(u, c):
    return u - u**3/3 + c


def plot_activation_function():
    fig, axs = plt.subplots(1, 1, constrained_layout=True, figsize=(16,9))
    counter = []
    for i in np.linspace(0.0,4.0,200):
        c = i
        counter.append(spike_count(np.transpose(
            euler(fitzHugh_Nagumo,0.1,1000,[-3.0/2.0,-3.0/8.0,i], 0)[0])[0]))
    axs.plot(np.linspace(0,5.0,200), counter, lw=1.0)
    cplot(axs, xlabel=r'$I$[nA]',ylabel=r'$\nu$[Hz]',
            title=r'Spiking Rate over Input Current', grid=True)
    fig.savefig('activation_function.png')
    fig.clf()

def plot_nullcline_trajectory(u_0, w_0, c, s):
    fig, axs = plt.subplots(1, 1, constrained_layout=True, figsize=(16,9))
    u = np.linspace(-3,3,800)
    axs.plot(u,lin_nullcline(u), lw=1.0, label=r'Linear Nullcline')
    axs.plot(u,cub_nullcline(u, c), lw=1.0, label=r'Cubic Nullcline')

    x, t = euler(fitzHugh_Nagumo, 0.1, 500, [u_0, w_0, c], 0)
    u, w, i = np.transpose(x)
    axs.plot(u,w, lw=1.0, label=r'Trajectory for $\nu$ '+s+' 0')
    cplot(axs, xlabel=r'$u$[mV]',ylabel=r'$w$[nA]',
            title=r'Nullclines and Trajectory for $\nu$ '+s+' 0', grid=True)
    fig.savefig('nullcline_trajectory_'+s+'.png')
    fig.clf()
```

```python

if __name__=='__main__':
    plot_activation_function()
    plot_nullcline_trajectory(1.0, 1.5, 0.6, 'equal')
    plot_nullcline_trajectory(1.5, 2.0, 2.0, 'unequal')
```