Week 2: Single Neuron Models

Phong Le, Willem Zuidema

November 6, 2013

In today's computer lab we will have a closer look at two differential equation models of single neuron dynamics: the Fitzhugh-Nagumo model and the Izhikevich model. We start with a very brief introduction to dealing with vectors in R and plotting your results, and then look at how we can use a package for analyzing (linear) systems of differential equations.

Note: if you are familiar with R you may skip the section 1.

1 Basic R: vector and graphics

If you are absolutely new to R, you might want to first go through the first three pages of this R tutorial:

- `http://www.illc.uva.nl/LaCo/clas/clc13/assignments/deboer13rtutorial.pdf`

Ready? Then, say hello to R by typing `print("Hello R!")`.

1.1 Vector

There are many ways to construct a vector. The first, also the simplest, way is to use the `c` function, e.g., `vec <- c(1,2,3)` creates a *column* vector

$$vec = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Another way is to use the `seq(n,m,by=k)` function, which creates $(n, n+k, n+2k, ..., n+uk)^T$ vector (where $u = (m-n)$ div k). For instance, the output of `seq(1,1.5,by=0.1)` is the vector $(1, 1.1, 1.2, ..., 1.5)^T$.

> Exercise 1.1: Create the following vectors
>
> $$a = \begin{pmatrix} 1.5 \\ -1 \\ 3 \end{pmatrix} \quad b = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 4 \\ 6 \\ ... \\ 24 \end{pmatrix}$$

> Exercise 1.2: Given vectors in the above exercise,
>
> - What do you get with `a + b`, `a - b`, `a * b`, `a / b`?
> - What do you get with `a > b`, `a < b`, `a == b`, `a != b`?

We can get an element by using the operator `[]`. For instance, `vec[i]` points to the i-th element of a vector `vec`, The operator `[]` can do more than that: we can get a set of elements. For instance, `a[c(1,3)]` and `a[c(TRUE,FALSE,TRUE)]` point to the first and the third elements of vector `a`.

Exercise 1.3: Find all positive elements in `a`. (Hint: use `a > 0`.)

1.2 Graphics

Line plot In order to draw a function $y = f(x)$, we can use `plot(x,y,type='l')` where `x` is a vector containing n ordered values of x and `y` is a vector such that `y[i] = f(x[i])`. For instance, the following code will draw the function $y = sin(x)$

```
1 x <- seq(-5,5,by=0.1)              # create x = (-5,-4.9,-4.8,...,4.9,5)
2 y <- sin(x)                        # compute y values
3 plot(x,y,type='l',col='blue')      # draw the graph, using blue color
4 savePlot("sin.png",type="png")     # save the current plot to a png file
```

If you want to draw another function on the same graph, you need to use the `lines` function. For instance, insert `lines(x,cos(x))` before the last line. Now, you should get a graph similar to Figure 1.

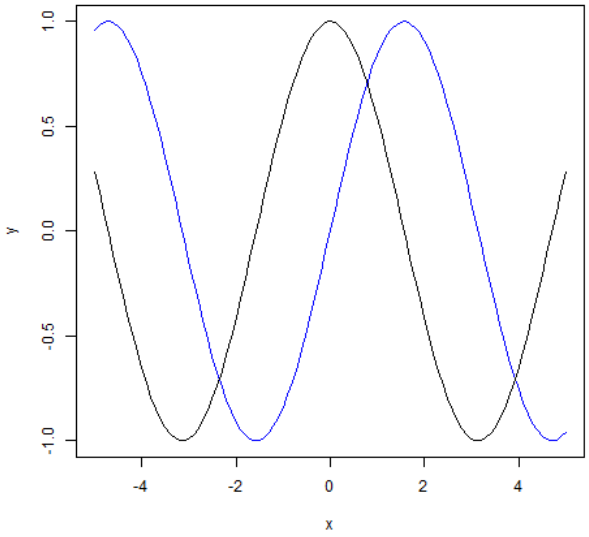

Figure 1: Drawing two functions, $y = sin(x)$ and $y = cos(x)$ in one graph.

Scatter plot Remove `type='l'` and replace `lines(x,cos(x))` by `points(x,cos(x))`, what do you get?

2 ODE and Phase Plane Analysis

Required Libraries There are two libraries we need for the task, `deSolve` for solving ODEs, and `fields` for drawing vector fields. Install them by typing

```
1 install.packages("deSolve")
2 install.packages("fields")
```

If you install a library on a computer where you don't have administrator rights, e.g., a university computer, you need to use `install.packages("library_name", lib="your_lib_path")` where 'your_lib_path' is any directory where you have rights to write files. In order to use a library, execute `library(lib_name)`, for instance execute `library(deSolve)`.

Required R Code At `http://www.illc.uva.nl/LaCo/clas/fncm13/assignments/computerlab-week2/` you can find the four R-files you need for this exercise:

- 'draw.R' contains functions for drawing phase planes and trajectories,
- 'linear_ode.R', 'FN_ode.R' and 'I_ode.R' contain R code for linear models, the Fitzhugh-Nagumo model, and the Izhikevich model.

You will be asked to add your own R code to those files.

2.1 ODE

A system of first-order ODEs we are interested in is defined as follows

$$\begin{aligned}\frac{du_1}{dt} &= f_1(t, u_1, ..., u_n, p_1, ..., p_m)\\ \frac{du_2}{dt} &= f_2(t, u_1, ..., u_n, p_1, ..., p_m)\\ &...\\ \frac{du_n}{dt} &= f_n(t, u_1, ..., u_n, p_1, ..., p_m)\end{aligned}$$

where t is an independent variable (e.g., time), $u_1, ..., u_n$ are dependent variables (e.g., variables describing the state of a dynamic system), and $p_1, ..., p_m$ are parameters.

2.2 Phase Plane Anaylysis

In this section, we will consider a system of two linear first-order ODEs

$$\begin{aligned}\frac{dx}{dt} &= ax + by\\ \frac{dy}{dt} &= cx + dy\end{aligned}$$

where x, y are dependent variables, and a, b, c, d are parameters. This system has two linear nullclines:

$$\begin{aligned}\frac{dx}{dt} &= ax + by = 0\\ \frac{dy}{dt} &= cx + dy = 0\end{aligned}$$

The intersection of the two nullclines, if it exists, is called *equilibrium*.

Now, open the file 'linear_ode.R', you should find the following function which declares the system of ODEs above:

```
# input:
#   t: current time point
#   var: c(x,y) current state
#   p: params, c(a,b,c,d)
# output: list(c(dx/dt,dy/dt))
linear.ode.system <- function(t, var, p) {
    x <- var[1]; y <- var[2]
    a <- p[1]; b <- p[2]; c <- p[3]; d <- p[4]

    return(list(c(a*x + b*y, c*x + d*y)))
}
```

and the following function for nullclines

```
# give x, find y such that
# dx/dt = dy/dt = 0
linear.nullcline <- function(x, p) {
    a <- p[1]; b <- p[2]; c <- p[3]; d <- p[4]

    return(c(-a/b*x, -c/d*x))
}
```

Exercise 2.1: Check for yourself how these functions work.

Now also look at the functions draw.phase.plane and draw.trajectory.over.time in the file 'draw.R'. Note that you don't need to fully understand the bodies of the functions, just skim them to get an idea how they work. Execute the code in draw.R (by using the menu or by typing `source(file.choose())` in the console and selecting draw.R).

To get an idea what the arguments are, let's execute 'linear_ode.R'. To do that, change your working directory in the menu to the folder where linear_ode.R is and then execute it with `source("linear_ode.R")`.

```
1  ######### linear ode ############
2  test.ode.system = linear.ode.system
3  test.nullcline  = linear.nullcline
4  test.params     = c(4,-3,15,-8)
5  test.xrange     = c(-10,10)
6  test.yrange     = c(-10,10)
7  test.xstep      = 1
8  test.ystep      = 1
9  test.xlabel     = "x"
10 test.ylabel     = "y"
11 test.x.ini      = c(2)
12 test.y.ini      = c(-10)
13 test.times      = seq(1,5,by=0.01)
14 test.ode.root   = NULL
15 test.ode.event  = NULL
16
17 ############ run experiment ##########
18 draw.phase.plane(
19              ode.sys = test.ode.system,
20              ode.event = test.ode.event, ode.root = test.ode.root,
21              nullcline = test.nullcline,
22              params = test.params,
23              xrange = test.xrange, yrange = test.yrange,
24              xstep = test.xstep, ystep = test.ystep,
25              xlabel = test.xlabel, ylabel = test.ylabel,
26              x.ini = test.x.ini, y.ini = test.y.ini, times = test.times)
27
28 draw.trajectory.over.time(
29                          ode.sys = test.ode.system,
30                          ode.event = test.ode.event, ode.root = test.ode.root,
31                          params = test.params,
32                          x.ini = test.x.ini, y.ini = test.y.ini,
33                          times = test.times,
34                          xlabel = test.xlabel, ylabel = test.ylabel)
```

The first four arguments (lines 19-21) are to tell which system of ODEs we are interested in. The fifth argument (line 22), `params = c(4,-3,15,-8)`, says that the parameters for the system are `a = 4, b = -3, c = 15, d = -8`. Some of the other arguments are described in Figure 2.

Exercise 2.2: With the system given in Figure 2 and 3,

- Show that (0,0) is an equilibrium.

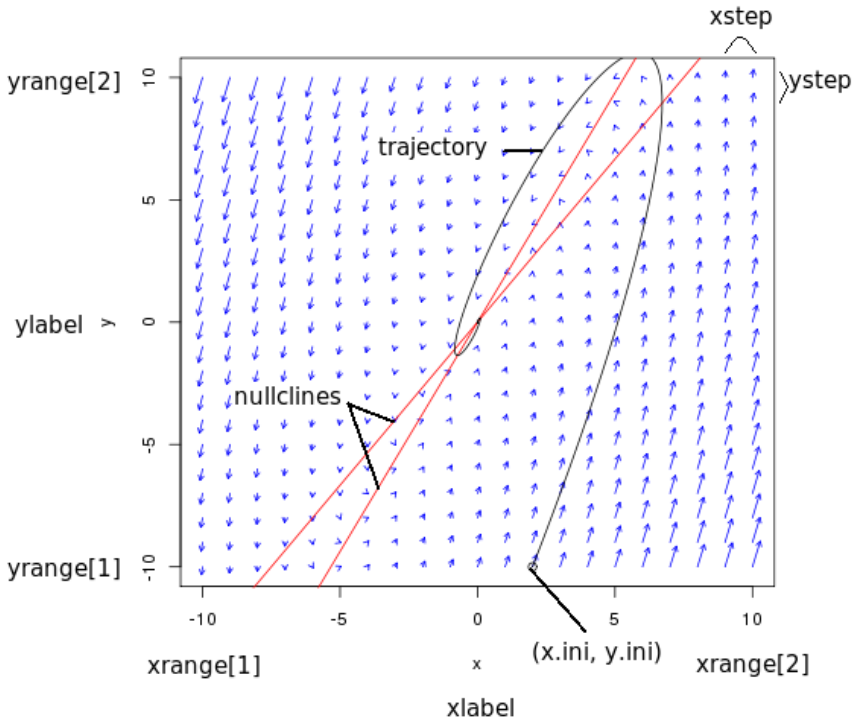

Figure 2: Phase plane of the system of two linear first-order ODEs with `a = 4, b = -3, c = 15, d = -8`.

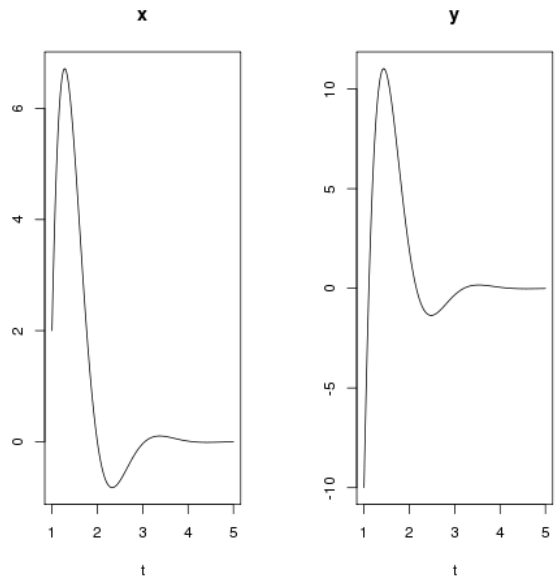

Figure 3: x and y versus t.

- Explain why, if the initial condition is a point very close to (0,0), the system will definitely end up at that point.

Exercise 2.3: Modify the file 'linear_ode.R' to draw the phase plane of the system

$$\frac{dx}{dt} = x + y$$
$$\frac{dy}{dt} = 4x + y$$

with the two initial conditions (4.8,-10) and (5.1, -10). (Tip: you can plot multiple trajectories in the same graph by letting text.x.ini and text.y.ini contain a list of x- and y-coordinates. E.g.,

```
test.x.ini = c(-3,3,1.2)
test.y.ini = c(3,-3,-0.9)
```

)

- Explain why the two trajectories have those shapes.
- Find an initial condition such that the trajectory ends up at the equilibirum.

3 Fitzhugh-Nagumo Model

In this section, we will explore the Fitzhugh-Nagumo model, which is described by the following system of two ODEs

$$\frac{dv}{dt} = c(v - \frac{1}{3}v^3 + r + I)$$
$$\frac{dr}{dt} = -\frac{1}{c}(v - a + br)$$

Exercise 3.1[1]: (Use the file 'FN_ode.R'; to change the parameters, you will need to change the line `test.params = c(0.7,0.8,3,0)`. Tip: put a `#` in front of test.params in FN_ode.R. You can then type `test.params = c(0.7,0.8,3,-0.2)` etc. in the console, and run the script with `source("FN_ode.R")`.)

- Draw the phase plane of the Fitzhugh-Nagumo model with $a = 0.7, b = 0.8, c = 3, I = 0$. Is the equilibrium point stable (i.e., are trajectories attracted to this point or repelled from it)?
- Try out different initial values for the trajectories (by changing text.x.ini and text.y.ini).
- Try 'depolarizing' the neuron from its equilibrium value at $(v, r) = (1.1994, -0.62426)$. What happens to the trajectories starting from $v = 1, 0.5, 0.0, -0.5$? I.e., try:

  ```
  test.x.ini = c(1, 0.5, 0.0, -0.5)
  test.y.ini = c(-0.62,-0.62,-0.62,-0.62)
  ```

- Change the injected current value to $I = -0.2$, the initial condition is $(v, r) = (1.1994, -0.62426)$. Is this point still stable?
- Determine what v versus t looks like for a trajectory on this phase plane. Would you classify the injected input of -0.2 as a superthreshold or subthreshold stimulus? Does this neuron exhibit subthreshold oscillations for this value of injected current?
- Change the injected current value to $I = -0.4$ with the initial condition $(v, r) = (1.1994, -0.62426)$. Is this point still stable? Plot several trajectories on this phase plane. Since the nullclines intersect at only a single point, there are no other equilibrium points for this system, but trajectories may be attracted to some other closed orbit: a limit cycle. Is there such a limit cycle in this system?

- Finally, repeat the analysis for $I = -1.6$ and examine v versus t. Does this neuron spike continuously as it did before? Neurons are known to exhibit a phenomenon called excitation block, whereby increasing the current injection can often repress repetitive firing behavior.

4 Izhikevich Model

In this section, we will explore the Izhikevich model, which is described by the following system of two ODEs

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$
$$\frac{du}{dt} = a(bv - u)$$

with the reset condition: if $v \geq 30$ then $v \leftarrow c$; $u \leftarrow u + d$. Because of the reset condition, the model turns out discontinuous at the reset point, and modelling it as an input to `deSolve` is a bit tricky. Therefore, in this section, we give you Izhikevich model code, and your task is simply try out some parameter sets.

Exercise 4.1: (Use the file 'I_ode.R') Plot the phase space of the model with parameters $(a, b, c, d, I) = (0.02, 0.2, -65, 8, 0)$, with null clines and a couple of trajectories that start from various points around the equilibrium. Do you observe qualitatively similar behavior to the Fitzhugh-Nagumo model with $I = 0$?

Exercise 4.2[2]: (Use the file 'I_ode.R') The purpose of this exercise is to see if you can discover what parameter sets lead to regular spiking, fast spiking, or intrinsically bursting behavior. What kinds of behaviors do the following parameter sets produce? (with $I = 10$)

- $(a, b, c, d, I) = (0.02, 0.2, -65, 8, 10)$
- $(a, b, c, d, I) = (0.02, 0.2, -55, 4, 10)$
- $(a, b, c, d, I) = (0.1, 0.2, -65, 2, 10)$
- $(a, b, c, d, I) = (0.1, 0.25, -65, 2, 10)$

(You may need to check this `http://www.izhikevich.org/publications/spikes.htm`.)

5 Submission

You have to submit a file named 'your_name.pdf' for those exercises requiring explanations and math solutions through Blackboard before 15:00 Monday 11 Nov.

[1]This exercise is the project in Chapter 12, "Exploring the Fitzhugh-Nagumo model", in the book Wallisch et al. "MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB".

[2]This exercise is Exercise 22.1 in Chapter 22, "Simplified Model of Spiking Neurons", in the book Wallisch et al. "MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB".