# CPIB SUMMER SCHOOL 2011: INTRODUCTION TO BIOLOGICAL MODELLING

## Practical 4: Spatial Models in MATLAB

Nick Monk

## 1 Getting started

Matlab files for this practical (Mfiles, with suffix '.m') can be found at:
http://www.maths.nottingham.ac.uk/mathsforlife/cpib2011
**As before, please ensure that you save all your files from this course in a separate directory, so that participants on the course do not accidentally overwrite each others work.**

## 2 MATLAB basics

MATLAB is a contraction of *MATrix LABoratory*, and as such it is principally a tool for numerical computation with vectors and matrices. In addition, it provides a high level programming language and advanced graphics, image analysis and visualisation tools. There is also a range of toolboxes providing specialised capabilities for problems in areas such as optimization, statistics, neural networks, image processing and even systems biology. Commands can be entered on the *Command Line* or stored as *scripts* (also known as *Mfiles*, simply a collection of MATLAB commands) or functions (which can be passed and return *arguments*) for execution on demand.

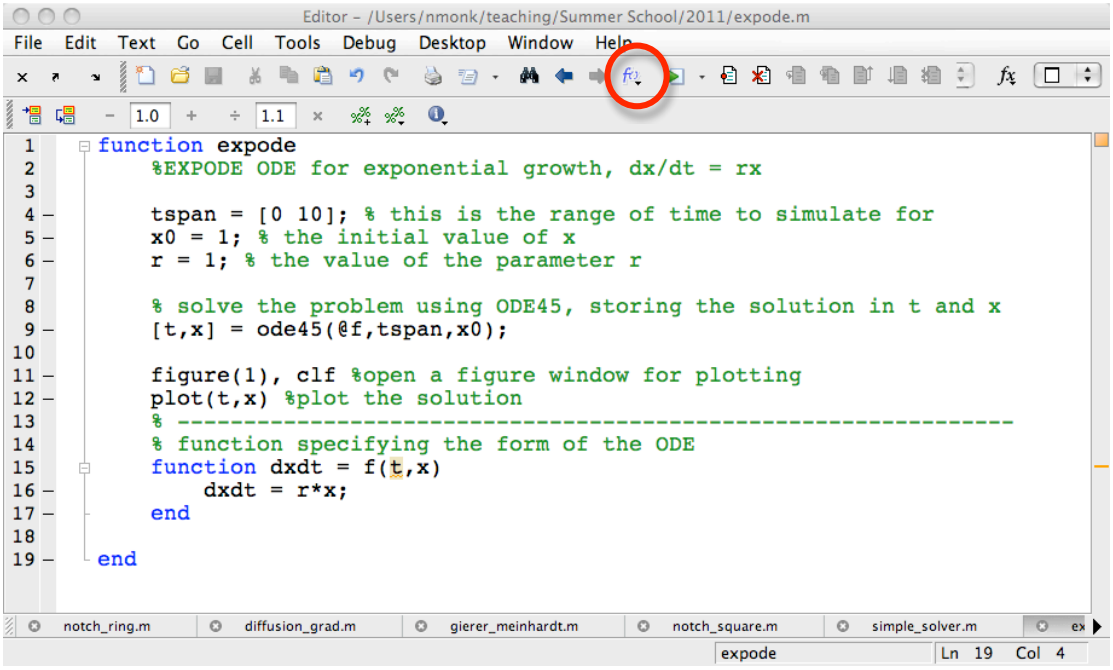
On University of Nottingham computer labs, you should be able to launch MATLAB via the "Start" menu:
Start > All Programs > UoN Software > Statistics and Mathematics >Matlab >MATLAB R2008a

Once you have launched MATLAB you should see a number of windows (or one window subdivided into sections). In the primary window containing the "Current Folder" item, set the Current Folder to be the one in which you have saved your Mfiles.

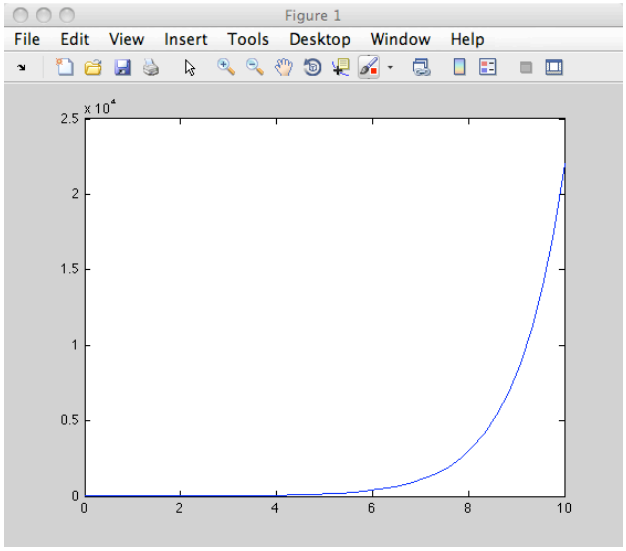
When you open Mfiles (using File > Open), they will appear in an Editor Window. This is just a text editor that has some Matlab-specific features (like automatic syntax checking and indenting). It is in this window that you should edit and save Mfiles.

There is also a Command Window (which has a >> prompt) in which you can enter commands interactively, should you wish. This is also the window where any error messages will appear if one of your Mfiles fails to run. These messages will usually tell you where in your Mfile the problem lies (or at least where the first encountered problem lies…)

We are using Matlab in this practical session because it is very easy to solve large systems of ODEs for spatial models, and to plot the results. To see the basics of how to solve ODEs in Matlab, download and open the Mfile `expode.m`. The code listing is shown below:




To run the Mfile, just click on the "run" arrow (highlighted in the above by the circle). This should open up a new plot window like the one in Fig. 1.




This is very basic, so as to focus on the way in which we can specify and solve ODEs in Matlab. To get a better idea of the kind of things we can do in Matlab, download the Mfile `simple_solver.m`. This function solves a two component ODE model representing a single gene feedback circuit, with *m* and *p* representing mRNA and protein, respectively:

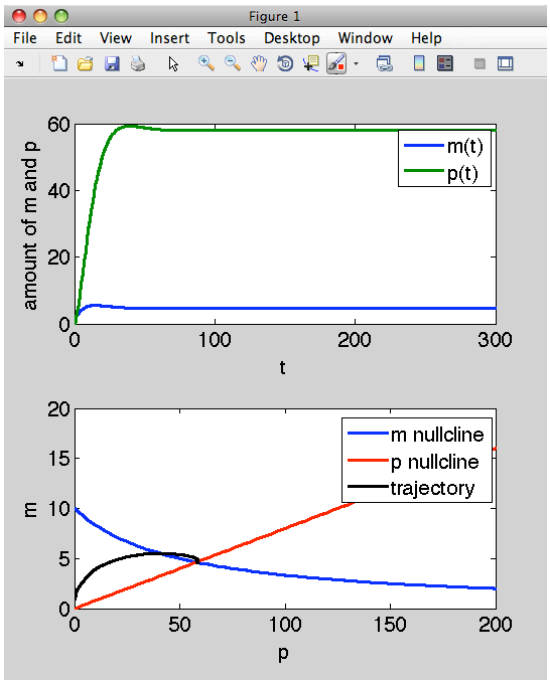
$$\frac{dm}{dt} = f(p) - d_1 m$$

$$\frac{dp}{dt} = k_2 m - d_2 p$$

*f(p)* is the function representing transcription. In the Mfile, you can select between an increasing and decreasing Hill function form for *f* by setting the parameter *a* (0 for increasing and 1 for decreasing).

In addition to a few tricks to make the plots look pretty, the Mfile contains a small routine that plots the nullclines for the system, and superimposes the system trajectory. This makes it easy to see what effect any changes in parameter values and/or initial conditions have on the nullclines and steady states, and on the model solution.

Run the Mfile without modification. You should get a plot window like the one below:




The upper plot shows the time course solutions for m and p, while the lower plot shows the nullclines and trajectory (spiraling in to the unique steady state). You may need to resize the window for best effect. You can use the magnifying glass icons to zoom in or out in either of the plots — just select the tool and then click on the region of plot you want to zoom.

**Revision Exercise**

Use `simple_solver.m` to explore positive feedback ($a = 0$). By setting $n = 2$, m_0 = 0, and p_0 = 42, you should see bistability, and also see that the system spends a long time in the vicinity of the unstable steady state before approaching a high ('on') steady state. Play around with the parameters and initial values a little to see if the behaviour seen in the time courses matches your intuition from the nullclines.

**Spatial Models**

The following Mfiles are provided:

1. notch_ring.m: Delta-Notch model in a ring of cells
2. notch_square.m: Delta-Notch model on an array of square cells
3. diffusion_grad.m: Diffusion of a morphogens in a ring of cells
4. gierer_meinhardt.m: 1D reaction-diffusion

Download these files and save them in your personal directory.

**1. Delta-Notch signalling in a ring of cells.**

The Mfile `notch_ring.m` (with the parameter `delay` = 0) solves the equations

$$\frac{dN_i}{dt} = p_N f\left(\langle D\rangle_i\right) - \delta_N N_i \qquad \langle D\rangle_i = \frac{D_{i-1} + D_{i+1}}{2}$$
$$\frac{dD_i}{dt} = p_D g\left(N_i\right) - \delta_D D_i \qquad f \text{ increasing}$$
$$g \text{ decreasing}$$

on a ring of $n$ cells (so that cell $n$ signals to cell 1, and vice versa), using Hill functions for $f$ and $g$.

1.1 Run the file for the default parameters. You should first get a window showing the time evolution of the Notch and Delta activities in the ring of cells; you should see a spatial pattern emerging over time. You may need to resize this for best results. In addition, when this has finished (it may take a little while), you should get two extra windows: one containing space-time plots of the solutions and one containing the nullclines for a two-cell system with fast Notch kinetics (see lecture for an explanation of this).

1.2 Increase `thresh2` to see what effect this has on the nullclines. You can speed this up by commenting out lines 40–42. What effect does this have on patterning?

1.3 Explore what effect the other parameters have on the system by varying one at a time (keeping `delay` = 0).

1.4 The parameter `delay` is a time delay. By setting `delay` > 0, we are simulating the model

$$\frac{dN_i}{dt} = p_N f\left(\langle D\rangle_i\right) - \delta_N N_i$$
$$\frac{dD_i}{dt} = p_D g\left[N_i\left(t-\tau\right)\right] - \delta_D D_i$$

which has delayed repression of Delta in response to Notch activity. Choose a parameter set for which the non-delayed model patterns, and explore the effect of increasing `delay` from 0 to 2. You may need to increase `tend` (the duration of the simulation). Is the final pattern affected?

**2. Delta-Notch signalling in an array of square cells.**

`Notch_square.m` solves the same model as above but on an array of square cells (so now $D$ is averaged over the *four* nearest neighbours of each cell in the Notch equation. To avoid boundary effects on patterning, periodic boundary conditions are applied (the array wraps round on itself). Two plot windows are produced: one showing the dynamics of Notch activity on the array of cells, and the other showing the nullclines as in the previous example.

2.1 Run the model and compare the behaviour to the ring case. You shouldn't find anything significantly different to the behaviour observed in a ring of cells (demonstrating either the remarkable "robustness" of the model behaviour, or a remarkable lack of versatility of the model, depending on your perspective!)

**3. Morphogen gradient formation**

`diffusion_grad.m` models the diffusion of a morphogen through a ring of cells. Morphogen is produced in just one cell in the ring, at a constant rate (given by the `source` parameter), and then

moves by random diffusion and undergoes degradation. The corresponding mathematical model is

$$\frac{dC_i}{dt} = D\left(C_{i-1} + C_{i+1} - 2C_i\right) - \delta C_i + \text{source},$$

where $C_i$ is the concentration of morphogen in cell $i$, $D$ is the diffusion coefficient, and $\delta$ is the morphogens degradation rate.

3.1 Explore how the steady distribution of morphogen depends on the parameters. Can you spot a correlation between the response to changes in $D$ and $\delta$?

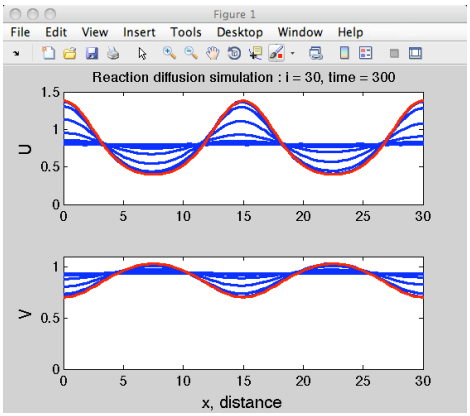
**4. Pattern formation by reaction-diffusion**

`gierer_meinhardt.m` simulates a reaction-diffusion system on a one-dimensional domain of a given length (this is a model parameter). The corresponding model is

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + a - u + u^2 v$$
$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + b - u^2 v$$

Like the Notch signalling model, this model can also generate pattern spontaneously (from small random initial perturbations to a spatially uniform state).

4.1 Run the model. The plot shows how the concentrations of $U$ and $V$ vary in time, with the current solution plotted in red, and past solutions plotted in blue (an alternative to the space-time plots we used in parts 1 and 3). You should see a spatial pattern emerge from the initial condition, which is close to spatially uniform (see below).




A possible interpretation of the steady spatial pattern of $U$ that develops is that it could act as a "prepattern" for cell fate/behaviour differentiation. The command window reports the results of calculations aimed at determining whether or not the parameters should support patterning (the Turing conditions).

4.2 If the Turing conditions are met, the model parameters determine the allowed pattern "wavelengths" (i.e. the distances between neighbouring activator peaks). Think about what the implications for patterning might be if the length of the domain is reduced or increased. Test your predictions by varying the model parameter `length` (found on line 16). You should find that patterning fails if the domain is too small. Can you find by simulation an approximate lower limit for the length? Can you think of how you might alter one or more of the other model parameters in order to restore patterning if the domain is below the critical length?