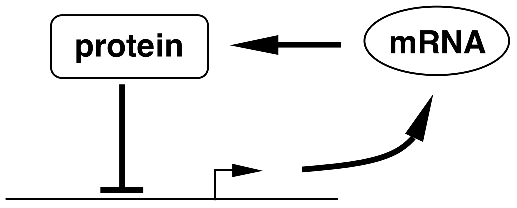
GSTDMB 2010:
MATHEMATICAL MEDICINE AND BIOLOGY
Practical 2.1: Multi-variable models in MATLAB

Nick Monk / Markus Owen

This set of practical exercises builds on those in Practical 1.1 by using MATLAB to simulate multi-variable ODE models. The MATLAB codes listed here can be found at: http://www.maths.nottingham.ac.uk/mathsforlife/gstdmb2010

1. A multi-component negative feedback loop

Many transcription factors negatively regulate their own production (by transcriptional repression of their own gene). A simple way of representing the state of this system considers two variables: mRNA concentration and protein concentration:



A simple model for this system represents four processes:

i) production of mRNA by transcription

ii) degradation of mRNA

iii) production of protein by translation

iv) degradation of protein.

Representing the concentration of mRNA and protein by m and p respectively, an ODE model for this system is:

$$\frac{dm}{dt} = \frac{k_1}{1+\left(p/p_0\right)^n} - d_1 m$$

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

where k_1, k_2, d_1, d_2 are the maximal transcription rate, translation rate, mRNA degradation rate and protein degradation rate, respectively.

i) To find the steady states of this model, set the rates of change of mRNA and protein to zero, giving:

$$0 = \frac{k_1}{1+(p/p_0)^n} - d_1 m$$
$$0 = k_2 m - d_2 p.$$

The second equation just says that, at steady state, $m = (d_2/k_2)p$. Substituting this in the first equation gives a single equation for the steady state level of p:

$$0 = \frac{k_1}{1+(p/p_0)^n} - \frac{d_1 d_2}{k_2} p.$$

The MATLAB function in "`ssplotter.m`" (you should have downloaded this as part of the archive "`prac-multi-odes.zip`") uses `fplot` to plot the function on the right hand side of this equation. Steady state protein expression levels correspond to points where this function crosses the horizontal axis.

With the parameter values: $k_1 = 50, k_2 = 1, d_1 = 0.1, d_2 = 0.1, n = 10, p_0 = 10$, show that the system has only one steady state, and use the plot to find an estimate for the steady state protein level.

Try varying the model parameters to see if the number of steady states changes. Do you think it can change? Can you see why based on the equation above?

ii) To find solutions of the equation $f(x) = 0$, you can use

```
>> fzero(@f,xstart)
```

where **xstart** is the starting point for zero finding. The MATLAB function in "`ssfinder.m`" uses `fzero` to find a better estimate of the steady state protein concentration (i.e. of where the above function crosses zero).

For the parameters above, use "`ssfinder.m`" to find the steady state protein level. By using the steady state relation $m = (d_2/k_2)p$ find the corresponding steady state value of the mRNA.

iii) In order to explore the effect of the parameter p_0 on the steady state level of protein concentration, repeat part (i) by increasing p_0 from 5 to 30 in steps of 5. For each value, keep a record of the steady state value (e.g. by growing an array containing the values—see lecture notes). Use `plot` to plot the steady state concentration for p against p_0.

iv) A function (`negode.m`) for simulating this model is provided in the archive `prac-multi-odes.zip`. It uses the parameters above, initial conditions $m=1, p=5$, and simulates the model from $t=0$ to $t=100$. Run this function to simulate the model. What do you notice about the system behaviour for these parameters?

v) The plot generated by the function plots both the time-course of mRNA and protein expression on the same axes. Note that the concentration of mRNA is significantly lower than that of protein (as you would expect, since at steady state $m = (d_2/k_2)p = 0.1p$ for our parameter set). This can make it difficult to see the mRNA dynamics clearly. To overcome this, we can use `subplot` to open multiple axes in a single plot window.

Replace the plot command in negode.m with the following:

```
subplot(2,1,1) %1st axes
plot(t,x(:,1),'linewidth',2) %mRNA
xlabel('time')
ylabel('mRNA')
subplot(2,1,2) %2nd axes
plot(t,x(:,2),'linewidth',2) %protein
xlabel('time')
ylabel('protein')
```

Re-run the function to see how `subplot` works.

vi) Use the function to explore the effects of varying the other parameters in the model. E.g. change d_1 and d_2 to 0.75 and note the more prominent overshoot of the steady state:

2. Incorporating protein dimerisation

Many transcription factors have to dimerise in order to bind DNA (e.g. basic helix-loop-helix proteins). We can include this in the model by adding another variable to the model to represent protein dimers:

$$\frac{dm}{dt} = \frac{k_1}{1+\left(p_2/p_0\right)^n} - d_1 m$$

$$\frac{dp_1}{dt} = k_2 m - d_2 p_1 - 2k_D p_1^{\,2}$$

$$\frac{dp_2}{dt} = k_D p_1^{\,2} - d_3 p_2,$$

where m, p_1 and p_2 represent mRNA, monomeric protein, and dimeric protein, respectively. d_3 is the degradation rate of protein dimers. In this model, mass action kinetics are used to represent the dimerisation reaction, and dimerisation is assumed to be irreversible with rate k_D:

$$\mathrm{P}_1 + \mathrm{P}_1 \xrightarrow{k_D} \mathrm{P}_2$$

i) Using `negode.m` as a basis, write a MATLAB function `dimerode.m` to simulate this dimerisation model for a time-span $0 \le t \le 100$. Set k_D=1, $d_3 = 0.5d_2 = 0.05$. Keep the other parameters the same as listed in part 1. You can assume that the initial concentration of p_2 is 5. You will need to add an extra entry to the column vector specifying the ODEs, and another `subplot` command (note that subplot(3,1,2) opens a 3×1 array of axes and prepares to plot in the 2nd of these. The last entry (2)

can range from 1 to 3 to select which axes to plot in. See `help subplot` for more information).
What do you find with this relatively rapid dimerisation rate? Compare your results to the model without dimerisation.

ii) Decrease the dimerisation rate to k_D=0.1, 0.01, 0.001. What happens to the magnitude of the overshoot of the steady state? For k_D=0.001 try increasing the time-span of simulation by setting

```
tspan = [0 1000];
```

What do you notice? What happens as you decrease the dimerisation rate further? What happens to the frequency of any oscillations? Is this what you expected?

iii) With k_D=0.001, reduce the cooperativity n of the Hill function from 10 to 2 in steps of 1. Describe what happens to the time course of mRNA and protein. Is this what you expected? Can you make biological sense of this?

iv) The form of transcription rate used assumes that the transcription factor can reduce the rate of transcription to zero (for high concentrations). In reality, some residual transcription is still possible even when the concentration of repressor is high. This can be incorporated in the model by adding a basal transcription rate k_0:

$$\frac{dm}{dt} = k_0 + \frac{k_1}{1+(p_2/p_0)^n} - d_1 m$$

$$\frac{dp_1}{dt} = k_2 m - d_2 p_1 - 2k_D p_1^2$$

$$\frac{dp_2}{dt} = k_D p_1^2 - d_3 p_2,$$

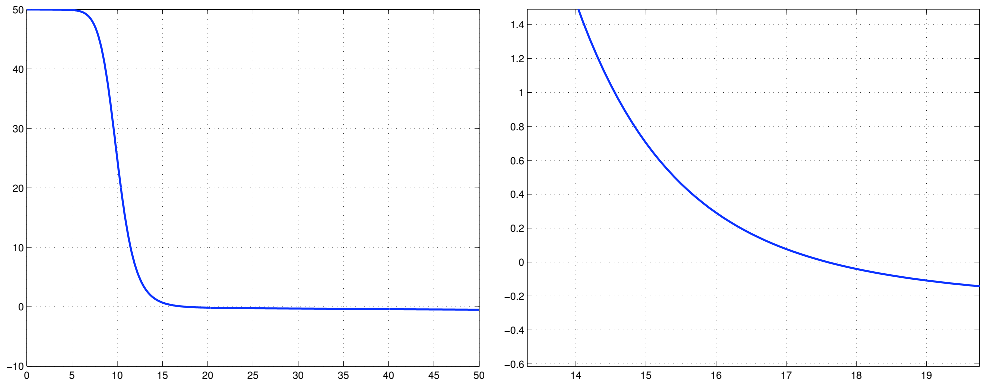
With k_D=0.001 and n = 10 as before, find out what happens to the oscillations as you increase k_0 from zero.

GSTDMB 2010:
MATHEMATICAL MEDICINE AND BIOLOGY
Practical 2.1: Multi-variable models in MATLAB
SOLUTIONS

Nick Monk / Markus Owen

1. A multi-component negative feedback loop

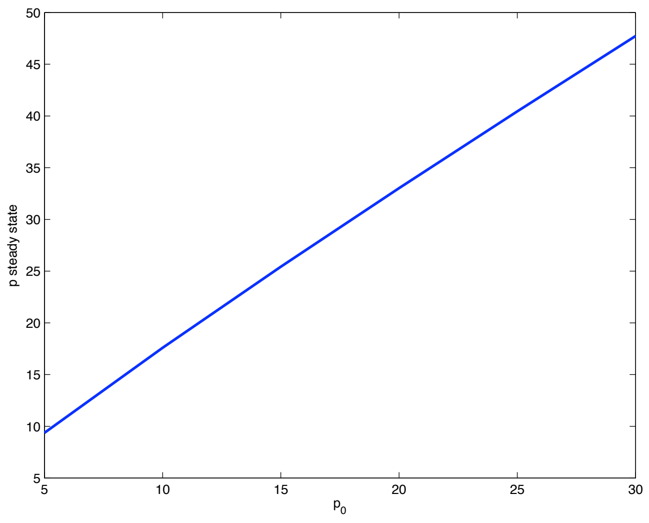
i) With the given parameter values, the plot generate by **`ssplotter.m`** is:



By zooming in we can estimate the steady state protein level as pss=17.5.

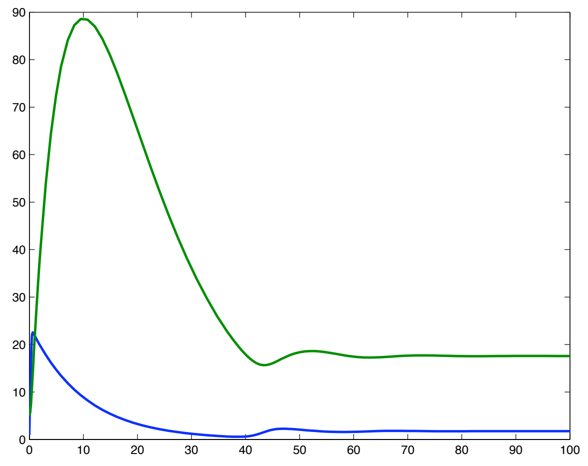
ii) The steady states given for these parameters by **`ssfinder.m`** are:
`pss=17.5882, mss=1.7588`

iii) The plot is shown below, generated by **`paramdep.m`**:



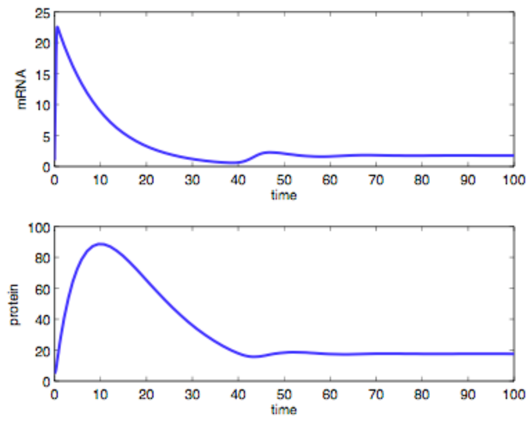
As the threshold for transcriptional inhibition is increased, the steady state protein level increases, which should fit with your intuition about this system.

iv) Using **`negode.m`** generates this output:

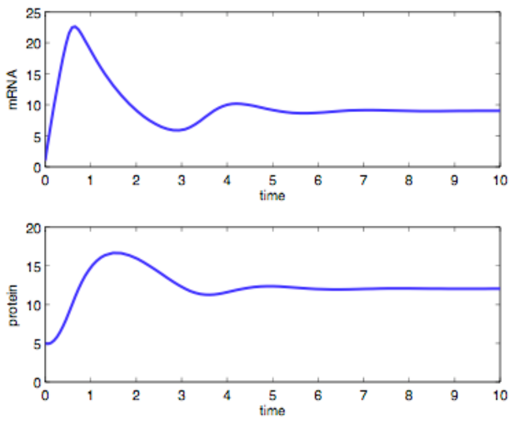


Notice how the solutions overshoot the steady state.

v) Using `subplot` we get the following output (see **`negodesubplot.m`**):

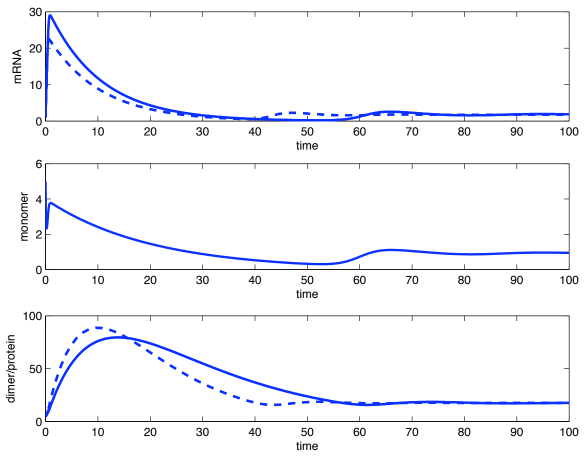


vi) Changing d_1 and d_2 to 0.75 we see a more prominent overshoot of the steady state (note the shorter time span to highlight the interesting dynamics – the steady state is reached much more rapidly for these parameters):

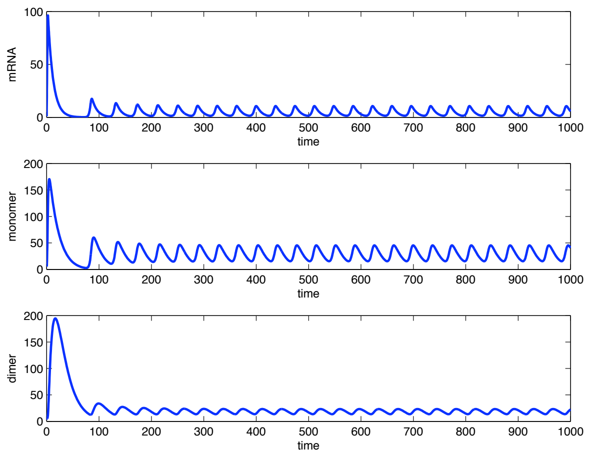


2. **Incorporating protein dimerisation**

i) A MATLAB script can be found in **`dimerodesol.m`**
The model with these parameters produces similar results to the model without dimerisation, with almost identical steady state levels of mRNA level and protein/dimer. Using **`dimerodecompare.m`** with **`negodecompare.m`** we superimpose the two simulations:

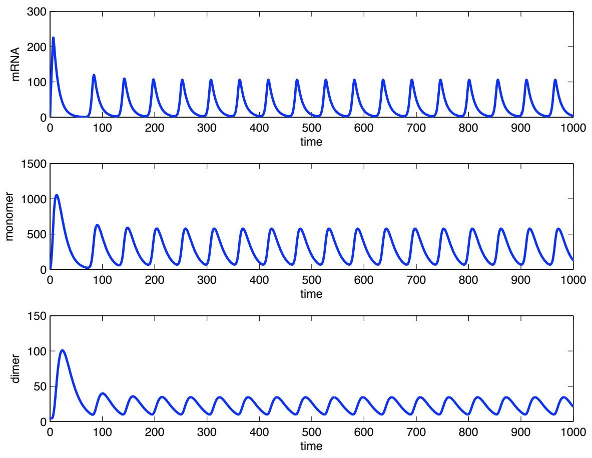


ii) As k_D decreases, the overshoot gets larger. When to k_D=0.001 we see sustained oscillations:



The system turns into an oscillator because the slow dimerisation step introduces further lag into the system. This causes the system to overshoot the steady state significantly. This example illustrates the importance of checking that any simplifying assumptions you have put in a model (like lumping monomeric and dimeric forms of a transcription factor into a single variable) don't radically affect the dynamical behaviour of the model for relevant parameters.

As you decrease the dimerisation rate further, the oscillations increase in amplitude and their frequency reduces (although not perhaps as much as you might expect). The figure below shows a simulation for k_D=0.00001:



In summary, smaller dimerisation rates give more pronounced oscillations because there is now even more lag in the system. For larger dimerisation rates, the oscillations die out because there is insufficient lag in the system to sustain them. You might not expect at first glance that increasing the dimerisation rate (promoting formation of the active form of the transcription factor) would reduce oscillations.

iii) As n is reduced, the oscillations die out. n gives a measure of the sensitivity of transcription rate to changes in the (dimeric) transcription factor concentration. For large values, just a small change in transcription factor gives a large change in transcription rate. This enhances overshoot of the steady state (and hence oscillations).

iv) A MATLAB script can be found in **`dimerodebasalsol.m`**.
As you increase k_0, you should again see the oscillations die out. Adding a basal transcription rate again reduces the sensitivity of the system around the steady state (this can be shown mathematically, as can all the other results).