CELL3006 Test 3:

**Synthetic Genetic Oscillators**

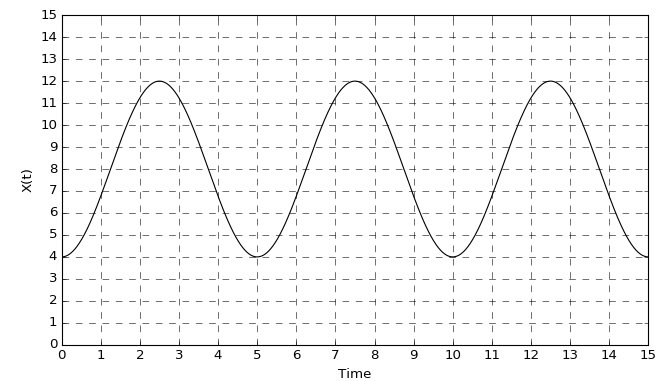
Assessed Practical 2

NAME:

**Part 1: Periodic Functions**

1.0) Look at the periodic function below.

Mark on arrows to indicate the period and amplitude of the oscillation.



The function above was generated by the following ODEs:

***Oscillator 1:***

A template for the simulating this model can be found in file oscillator\_type1.py

1.1) Open the file, and complete the code by entering the correct model equations.

The file includes values for initial conditions and parameter values. (Note that parameters k2 and k3 are not used here but will be needed later.)

1.2) Code to create a figure with two plots side-by-side has already been entered. Add code to plot the results of the simulation into the figure with:

* A plot of the time evolution of *X* vs. *t* on the left hand side.
* a phase plot showing the trajectory on the right hand side

e.g. *V* on y-axis vs. *X*  on x-axis

1.3) The behaviour of variable X is periodic. Write down the time period, frequency and amplitude (assume time units are seconds).

1.4) How does the phase plot you have drawn indicate that the oscillations are sustained?

1.5) Edit your code to add a second trajectory onto both plots showing the results of a simulation starting at initial conditions , .

How has change in initial conditions changed the behaviour of *X*?

**FIGURE OUTPUT: oscillator1.png**

1.6) Is this ODE model a harmonic or limit cycle oscillator? Explain your answer.

1.7) Let’s say we made an adjustment that changed the phase of the second oscillator. Describe how would this change the plotted graph?

Open the file oscillator\_type2.py (note that it is supposed to be empty!)

Copy your code into it.

1.8) In this file adjust the model equations according to the expressions below:

***Oscillator 2:***

**FIGURE OUTPUT: oscillator2.png**

1.9) Describe the behaviour of this system, including a comparison to the behaviour of oscillator 1.

Open the file oscillator\_type3.py. Copy your code into it.

1.10) In this file adjust the model equations according to the expressions below. You should find the model still oscillates (if not check your equations carefully!):

***Oscillator 3:***

**FIGURE OUTPUT: oscillator3.png**

1.11) Again describe the behaviour of this system, including a comparison to the behaviour of oscillator 1.

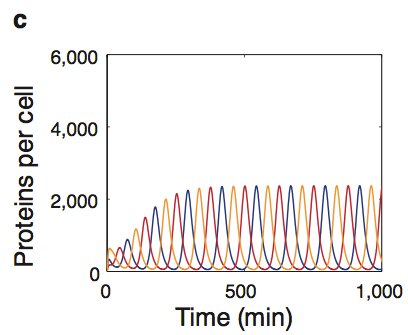
**Part 2: A deterministic model of the Repressilator**

***Plotting the behaviour***

Open the file repressilator.py

This contains code to run a deterministic simulation of the Repressilator.

The written code to run the simulation is complete, however the results are not yet plotted.

2.1) Add code so that the file creates a figure that reproduces the ODE simulation plot presented by Elowitz and Liebler:

Note the units used in our simulation are seconds. To convert the time values into minutes when plotting, you can use plot the simulation output using a command like the following:

ax.plot(t\_obs/60, p\_obs)

You do not need to use the same tick-marks on the axis scales, but should try to match the axes limits, and label the plot appropriately.

**FIGURE OUTPUT: repressilator.png**

2.2) Compare the observed behaviour to the simulation result presented in the paper.

2.3) Adjust the parameters to set and rerun the model.

How has this changed the simulation output?

Suggest the reason for this change with reference to the conditions necessary for limit-cycle behaviour.

***Before continuing adjust the parameters back so that .***

***Extending the model to include GFP***

We can extend the model so that it also includes the level of GFP in the cell.

This let’s us compare more directly with the measured fluorescence time series reported in the paper.

**The species variables and parameters needed are already in the modelled system, but the rate calculations are missing.**

To include the rates we will assume that rates for mRNA transcription and translation of GFP are the same as for repressors LacI, TetR and CI.

Additionally we will assume that the GFP mRNA degrades at the same rate as the mRNA for the LacI, TetR and CI repressor genes.

This means when constructing the rate equations we can reuse the form and parameters for protein/mRNA production and mRNA degradation.

We cannot do the same for GFP protein degradation/dilution as it has a half-life (90 minutes) that is much longer than that of the repressor proteins.

To add this into our model we must convert the half-life to a rate constant that can be used in a rate expression for degradation/dilution of :

2.4) Write down the calculation necessary to do this conversion.

You should find that a half-life of 90 minutes leads to the following rate constant

Hint. The above differential equation means:

2.5) Look at the Repressilator paper. Which repressor protein inhibits GFP production?

2.6) Using the assumptions detailed above, edit the code to complete the missing rates for:

rate\_m\_GFP\_prod

rate\_m\_GFP\_loss

rate\_p\_GFP\_prod

rate\_p\_GFP\_loss

2.7) Run the code and produce a new figure showing the behaviour of against time.

*If you don’t see oscillatory behaviour check you have reset .*

**FIGURE OUTPUT: repressilator\_GFP.png**

2.8) Comment on the modelled behaviour of , noting how it compares to:

i) the concentration of the repressor proteins

ii) the experimental fluorescence time series shown by Elowitz and Liebler.

**Part 3: A deterministic model of the Repressilator**

***Plotting model behaviour***

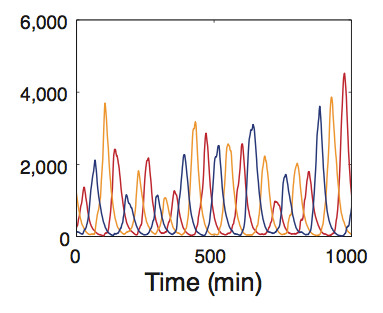
Open the file repressilator\_stoch.py.

It provides a framework to run stochastic simulations of the Repressilator genetic circuit (using the Gillespie algorithm).

This code is complete except for the section that updates the species states following event type selection.

3.1) Edit the simulation function to complete the algorithm code.

3.2) Add code so after it runs a single simulation ofthe model, it produces a plot that can be compared to the stochastic simulation presented by Elowitz and Liebler:



**FIGURE OUTPUT: rep\_stochastic.png**

3.3) Comment on how our model output compares to the authors stochastic simulation.

***Analysing pre-generated results***

The code in file repressilator\_analysis.py lets us load a set of data that has been generated previously.

The data file is called repressilator\_output.pickle. Ensure this is saved into your working directory so that Python can load it.

The data in the file was produced by creating simulating 100 runs of the the stochastic model and saving the output array produced.

The loaded array can be treated exactly the same as the output array produced by the multiprocess function, i.e. you can iterate over the runs using a loop like this:

for s\_obs in output:

ax.plot(t\_obs,s\_obs[:,0])

3.4) Write code that creates a figure overlaying the trajectories from the first two runs showing only the number of LacI proteins.

Comment briefly on what you observe.

**FIGURE OUTPUT: rep\_runs.png**

3.5) How it would compare to an equivalent plot produced overlaying two trajectories both simulated using the deterministic model

3.6) Write code to calculate the averaged trajectory of p\_LacI over all 100 runs, and add it onto the figure.

Comment on the behaviour shown, (and suggest a reason for the trend if you can think of one).

**FIGURE OUTPUT: rep\_runs.png (updated)**

3.7) Consider the processes included in the stochastic model.

What additional sources of stochasticity (noise) occur in the real system that are not included in this model?

3.8) The file repressilator\_analysis.py includes a function called peak\_finder that can scan over a timeseries and locate the position of the peaks.

For example, if we have a set of observations stored in arrays:

t\_obs and p\_LacI

we can call it using:

maxima = peak\_finder(t\_obs, pLacI\_obs,100)

Here the third argument (100) corresponds to a threshold which is the minimum height necessary for recording a peak.

The function returns a list: maxima. This is a set of coordinates containing the time and height for each peak identified.

3.8) Run the peakfinder function on the trajectory of p\_LacI recorded in the first stored run (i.e. corresponding to observation stored in output[0])

Examine the length of maxima to find how many peaks were detected, and

print the first recorded maximum.

What is the time and height of this first peak?

Comment whether this is consistent with your plot.

3.9) Write code to iterate over the maxima list and mark onto the plot the positions of the peaks detected onto the figure using ‘x’s.

**FIGURE OUTPUT: rep\_runs.png (updated)**

3.10) Create an empty list called heights.

Write a loop that runs the peakfinder function on each of the 100 runs, and adds the height of every peak detected into the heights list.

3.11) Produce a histogram showing the distribution of peak heights in the stochastic simulation.

Comment on how the mean peak height observed compares to that expected from the deterministic model.

**FIGURE OUTPUT: peakheights.png**

**Extension: Produce a histogram showing the distribution of the peak-to-peak intervals, and comment on the distribution found.**

**FIGURE OUTPUT: peakintervals.png**

**Part 4: A deterministic model of the Repressilator**

4.1) Comment on the relative merit of using deterministic models vs stochastic modelling in the repressilator system.

4.2) Contrast the agreement between modelling simulations and experimental observations detailed in the original repressilator paper.

4.3) Elowitz and Liebler detail in their paper the effort they made to develop a quantitative model of Repressilator behaviour. Comment on the extent to which they achieved this.