Lab 2: Oscillations in Delay Differential Equations

Along with sensitive negative feedback, delays are a critical ingredient in causing oscillations in dynamical systems. Without any delays, even the most sensitive feedback cannot produce oscillations in a system of differential equations.

Delays in a model can be either implicit (arising from the structure of the model, particularly the number of variables) or explicit. Explicit delays occur when a term in a differential equation is a function of the value of the state variable some time ago. For example, we might have \( x'(t) = 2x(t - 5) \), where \( x(t - 5) \) is the value of \( x \) 5 time units before the present time. Such equations, which explicitly include time delays, are called delay differential equations.

Solving Delay Differential Equations

In order to solve a delay differential equation, we have to find the value of \( x(t - \tau) \) time units ago, plug this value into the equation, and then integrate one time step, just as we did with Euler’s method. Sage does not have a built-in solver that can do this, so we will use one that was written specifically for this course. The function is available in your turn-in file.

**Exercise 1.** Evaluate the cell in your turn-in file that contains the `dde.solve` function. Nothing visible will happen when you evaluate this cell, since it only defines the solver function.

One of the features that makes this solver a little different from the one we’ve used before is its use of Python dictionaries to hold delay values. A *dictionary* is a collection of pairs that have the structure *key : value*. To make a dictionary, enclose a collection of such pairs in curly braces. For example, here is a dictionary that lists the planets and each planet’s position in the Solar System.

```python
>>> planets = {"Mercury":1, "Venus":2, "Earth":3, "Mars":4, "Jupiter":5, "Saturn":6, "Uranus":7, "Neptune":8}
```

To find the dependent variable value associated with a particular value of the independent variable, use square brackets.

```python
>>> planets["Earth"]
```
Exercise 2. Create a dictionary called `menu` that gives the price of ice cream as $4.95, cookies as $2 and candy as $1. (Leave the dollar signs off.) Then, check the price of cookies.

A Model of Respiration

This lab will focus on the Mackey-Glass model of respiration described in your text. (Similar equations are used to model delays in other physiological processes.)

The state variable in this model is the concentration of carbon dioxide in arterial blood, which we'll call $x$.

The body's rate of production of CO$_2$ is a constant, which we'll call $L$. The rate of excretion of CO$_2$ from the lungs is a sigmoidal function of its concentration in the blood. The function $f(x) = \frac{V_{max} x^n}{h + x^n}$, which you studied in the previous lab, does the job for sufficiently large values of $n$.

There is one complication in this model - a delay between gas exchange in the lungs and the effect on CO$_2$-monitoring neurons in the brain. In simple terms, it takes time for blood to get from the lungs to the brain. Therefore, the brain is responding not to the current CO$_2$ concentration in the blood but to the concentration some time ago. (In the body, this delay is on the order of 0.2 minutes.) Thus, the CO$_2$ excretion function really needs to be $f(x) = \frac{V_{max} x(t-\tau)^n}{h + x(t-\tau)^n}$.

The more CO$_2$ is in the blood, the faster it leaves the blood, so the excretion function should be multiplied by the product of $x$ and some constant $a$ to get the actual excretion rate. Thus, the overall equation is $x'(t) = L - \frac{V_{max} x(t-\tau)^n}{h + x(t-\tau)^n} a x(t)$.

The parameter values we will use are $L = 6$, $V_{max} = 80$, $a = 0.2$, $n = 5$, $h = 400$ and $\tau = 0.2$.

The syntax for the `dde_solve` function is `sol=dde_solve(equation, statevar, delayedvars, history, tmax, timestep)`, where `delayedvars` is a dictionary of variables that represent values of the form $x(t-\tau)$ and the delays associated with each, `history` is a function describing the state variable values before the simulation began, and `timestep` is the step size.
Exercise 3. Declare $x$ and a variable representing $x(t-\tau)$ as symbolic variables. Then, run the simulation for 100 minutes, assigning the output to a variable. Use 0 for the history function.

Exercise 4. The Mackey-Glass model simulates CO$_2$ concentration, but we are interested in breathing rate. The two are linked by the function $v(x) = \frac{V_{\text{max}}x^n}{h+x^n}$, the same function used in actually formulating the model. Use this function to convert your simulation results into breathing rate.

Exercise 5. Plot your results from the previous exercise using list_plot. Make sure the time axis is appropriately scaled. Describe what you see.

Exploring the Mackey-Glass Model

We have said that a delay is necessary to cause oscillations in the Mackey-Glass model, but how much of a delay? Is the mere presence of any delay sufficient to cause oscillations or is there a threshold?

Exercise 6. Simulate the model for at least three different values of $\tau$. Since all we’re interested in is the presence or absence of oscillations, you don’t need to convert the $X$ values to breathing rate. Which of these values of $\tau$ cause oscillations?

Exercise 7. Write an interactive that will allow you to manipulate $\tau$ in the model.

Exercise 8. Use the interactive from the previous exercise to approximate the minimum delay needed for persistent oscillations.

The other key ingredient in oscillations is a sensitive negative feedback loop. Again, we are interested in the question of how much sensitivity is necessary to produce persistent oscillations.
**Exercise 9.** “Sensitivity” in the Mackey-Glass model is embodied in the steepness of the sigmoid function controlling respiration. Which parameter controls this steepness? If you’re not sure, plot the function and manipulate parameters to check.

**Exercise 10.** Having determined the parameter to target, copy and modify the interactive you wrote in Exercise 7 to allow you to manipulate this parameter as well.

**Exercise 11.** In the interactive you wrote in Exercise 10, set $\tau$ to the minimum value you found previously and manipulate your sensitivity parameter to find the minimum sensitivity at which persistent oscillations appear.

Manipulating parameters together can have very different effects from manipulating them separately.

**Exercise 12.** Pick at least five values of $\tau$ and, for each one, approximate the minimum sensitivity necessary for persistent oscillations. Make a list of your findings.

**Exercise 13.** Plot the pairs of $\tau$ and sensitivity that you found in the previous exercise. Describe what you find.