MATH 1070, Introduction to Computational Science

Lab Session 6 – Systems of DEs in MATLAB

Background

In previous Labs you moved beyond simple calculations, equations, function calls and plotting to solving DEs for physics problems. Solutions for selected exercises from the last Lab are included at the end.

Today's lab

In this tutorial you will we extend the above techniques to chemical and biological applications using MATLAB's ODE45. In Lab 4 you solved a simple ODE, that describes simple exponential growth or decay:

$$\frac{dy}{dt} = a\,y(t) \qquad \ldots. (1)$$

We now consider the numerical solution of sets (pairs, triples) of differential equations (DEs). Problems addressed include: applications to chemical reactions; constrained population growth; and coupled DEs applied to predator-prey competitions in ecology.

Exercise 1. DEs for first- and second-order Chemical Reactions

a) Eq (1) also describes the uni-molecular decomposition of a single chemical species (eg. dinitrogen pentoxide N_2O_5) – now " a " is the rate constant, usually denoted by " k ". Such rate constants are determined by experimental observation of actual chemical reactions; k may be sensitive to temperature – many reactions go faster at elevated temperatures. The gas-phase chemical reaction exhibits first order (ie. linear) kinetics and so can be represented by:

$$N_2O_5 \xrightarrow{k} 2NO_2 + (1/2)\,O_2 \qquad \ldots.(2)$$

The concentration of reactant, written in chemistry texts at $[N_2O_5]$, has units (moles/litre) and can be represented as x(t).

The rate equation for the reaction, (8.2) can be written as the DE:

$$\frac{dx}{dt} = -k\,x(t) \qquad \ldots. (3)$$

For Eq. (2) experiments measure the rate constant as k = 1.8 (hr^{-1}) at 45C.
Convert this k to a value in SI units (ie. sec^{-1}) – this is simple arithmetic.
If the initial concentration of reactant is 0.2 (m/l), numerically calculate the time course of the reaction over 30 mins. Plot the results and accurately label the axes and the graph.

b) We consider now a second-order (non linear) chemical reaction – the thermal decomposition of nitrogen dioxide at high temperatures:

$$2\,NO_2 \xrightarrow{k} 2NO + O_2 \qquad (4)$$

The concentration of reactant now is y(t). This is a bi-molecular reaction (two reactant molecules are involved), so that the rate (y') is observed to be:

$$\text{rate} = k\,[NO_2]^2 \qquad (5)$$

If you think about it for a minute you'll notice that now the units of k are different. At 325 C the value of k for this reaction, (4) is

k = 0.53 (mole/lite/sec).

The DE for this case is:

$$\frac{dy}{dt} = -k\,y^2(t) \qquad \ldots. (6)$$

As in (a), numerically solve Eq (6) to calculate the time course of the reaction over say 10 mins. Take y(0) to be 2.0 (m/l). Again plot the results and accurately label the axes and the graph.

Show that a plot of $[NO_2]^{-1}$ vs time is linear – this is a standard technique used in chemistry to determine k from observations.

Exercise 2 – Constrained (non-linear) population growth & decay

Consider the growth of a population that is constrained by the carrying capacity of the habitat, N* - this is the maximum value that the number of individuals, N(t), can attain. An early model of this type was applied by P. F. Velhurst (1838) in modeling the population of Europe. The so-called Velhurst equation is a non-linear DE for constrained population growth:

$$\frac{dN}{dt} = a\,N(t)[1 - N(t)/N^*] \qquad \ldots. (2.1)$$

In numerical simulations mostly it is better to work with dimensionless variables – for the present case, a suitable choice is:

$$x(t) = N(t)/N^*. \qquad \ldots. (2.2)$$

Now x adopts values: $0 <= x(t) <= 1$. Thus eqn (2.1) becomes:

$$\frac{dx}{dt} = a\,x(t)[1 - x(t)] \qquad \ldots. (2.3)$$

We will apply this DE to the microorganism, Paramecium. Then the rate parameter is

$a = 1.016\ (day^{-1})$.

In Matlab, again set up an array of time values, as in Lab 4. Follow the growth of this bug for, say 2 weeks. You will need to set the parameter(s) and Initial Condition(s), eg: take

(i) x0 = 0.02; % initial condition, x(t = 0)

You also need to write the RHS of the DE in a function m-file. For this case it contains a command like: xprime = a*x(1-x) . Be sure to comment your code fully.

Parameters, like a, can be declared *global* in MatLab so that they can be passed to the x'(x-prime) function. Use the MatLab Runge-Kutta methods ode45, to solve the DE numerically.

Graphically display your results by using the MatLab *plot* command to plot x(t). Use the *xlabel*, *ylabel* and *title* commands to label the graph clearly for future reference

Repeat the calculation for a larger initial population, eg:

(ii) also try x0 = 0.1, 0.2 - is the growth profile sensitive to IC?

Try a longer range of t, to explore what happens.

Exercise 3 - Pair of interacting predator-prey species

Ex 2 applies to a single species. Now we consider a pair of interacting species – a classic problem is a predator-prey interaction, such as between lynx and hares in Hudson Bay. One model, proposed by Lotka & Volterra, uses a pair of coupled, non-linear DEs to model this situation (cf. lectures):

$$\frac{dN}{dt} = [\,a - b\,P(t)\,]\,N(t) \qquad \ldots. (3.4)$$

$$\frac{dP}{dt} = [\,-d + e\,N(t)\,]\,P(t) \qquad \ldots. (3.5)$$

where N(t) is the prey population and P(t) is the predator population. Look at the equations to understand what is happening:

- in eqn(3.4) the "-b" indicates that the prey population (N) declines as the predator population (P) increases;
- similarly, in eqn(3.5) the "+e" indicates that the predator population increases as there are more prey (increasing N).

For the initial conditions:

N(0) = 20
P(0) = 30,

And the parameters:

a = 1 , b = 0.02
d = 1 , e = 0.1

numerically calculate the time course of both populations.
Now the call to ode45 has the form:

[t,x] = ode45('xprime', [0,T], [N0; P0])

where x is now an array, the 1-st column of which is vector, N(t) and the 2-nd column of which is the vector, P(t); and T is the max time.
The RHS of the two DEs are in a function m-file called *name* that calculates xprime. Hint: this should contain:

```
name = [0;0];  % to indicate two columns of output

name(1) = ...
name(2) = ...
```

and any other commands that you need.

Plot the results (on the same graph) and label the axes and the graph.

Also produce the phase plot for the pair of populations – ie. a parametic plot of P(t) vs. N(t). What does this tell you about the stability of the populations and their interactions?

Explore the dynamics of this population system by varying some of the parameters.

Exercise 4 (additional) - Logistic iteration & Chaos

This section of the practical session is concerned with the numerical solution of non-linear difference equations, specifically the Logistic Equation – made famous by Lord Robert May (Nature, 1996 – see class handout). The main application is to population dynamics. This equation is limited to populations with discrete, ie non-overlapping, generations.

At generations n =1,2,3,.... the population N has discrete values N_1, N_2, N_1,.... The basic form of the Logistic iteration is

$$N_{i+1} = N_i (a - b N_i), \quad i = 1, 2, 3, ... \qquad .... (4.6)$$

Here a is the basic growth rate of the population and b describes a reduction in growth rate due to resource constraints. If b=0, then there is just unconstrained, exponential growth: $N_i = a^i N_0$, much like exponential growth when a > 1. By normalising N_i,

$$x_i = (b/a) N_i,$$

we have a dimensionless equation with $0 <= x_i <= 1$; and a new parameter $\mu = b/a$; this has the range of values (0,4).
Eqn (4.6) now has the simpler (canonical) form:

$$x_{i+1} = \mu x_i (1 - x_i), \quad i = 1, 2, 3, ... \qquad .... (4.7)$$

In this lab exercise we will explore the behaviour of Eqn(4.7) as μ is varied.

In Matlab, you will need to set the parameter value and Initial Condition, eg: take

(i) x1 = 0.2; % initial condition, x(i = 1).

Write a simple for loop to follow the Logistic iteration over some number of steps (eg. 20, 50, etc) – use as many as you need to get the "final" behaviour.

Then claculate the Logistic iteration numerically, as listed below. Graphically display your results by using the MatLab *plot* command to plot the sequence x_i. Use the *xlabel*, *ylabel* and *title* commands to label the graph clearly for future reference

1. One stable state: for $\mu < 3$, explore the behaviour of Eqn(4.7). Where does the transition to two stable states occur? Vary the initial condition to see what happens.

2. Two stable states: $3 < \mu < 3.6$. Note the oscillation between two, and then four, stable population values as μ is increased. Again check the sensitivity to initial conditions.

3. Chaos: increase μ slowly to, say, 3.9 and observe the wild oscillations in population values.

Solution: Ex 2.3 (Lab 4 – Kittinger free fall, with air drag)

```
% free fall under gravity, with air drag ~ Vel^2
% first: Create the function
function yprime=yvder_drag2(t,y)
m=150;
g=9.8;
k=0.408;
yprime=[y(2); g-(k/m)*y(2)^2];

% And in the main program, o call that fn:
time=[0:2500];
yvzero=[3904, 275.4882];
[t,y]=ode45('yvder_drag2', time, yvzero);
```

Solution: Ex 1 (Lab 5 - Plotting)

```
1/  % 1-D plots
a=12;b=5;
t=0:0.05:10*pi;
x=(a+b)*cos(t)-b*cos((a/b+1)*t);
y=(a+b)*sin(t)-b*sin((a/b+1)*t);

plot(t,x,'b',t,y,'g*')

2/
xlabel('x(t) and y(t)'), ylabel('t')
legend('x(t)','y(t)')

3/
figure
plot(x,y,'pr')
axis equal

4/
axis([-25 25 -25 25])
grid on
title('epicycloid: a=12, b=5')
xlabel('x(t)'), ylabel('y(t)')

8/
subplot(3,1,2)
plot(t,x,'b',t,y,'g*')
subplot(3,1,3)
plot(x,y,'pr')
axis equal
axis([-25 25 -25 25])
```

Solution: Ex 2

```
1/  % 2-D plots
[x,y] = meshgrid(-2.1:0.15:2.1,-6:0.15:6);
u =80.*y.^2.*exp(-x.^2-0.3.*y.^2);
mesh(x,y,u)
xlabel('x');
ylabel('y');
zlabel('u');
```

```
4/
surfl(x,y,u)
colormap gray
shading interp
```

Solution: Ex 3

```
% I/O basics
load -ascii mydata.txt
t=[1:size(mydata)]';
z=[data t]
save('new_data.txt','z','-ASCII')

% with overkill
%mat = fopen('mydata.txt', 'rt');
%[data] = fscanf(mat,'%e \n', [235]);
%fclose(mat);
```