LAB 1: CANCER, SERIES, AND ODE SOLUTIONS, PART A

1. Introduction

How should you work this lab? It has several sections, and a Part A and Part B. You should read through each Part before you start working on it. Note that both parts have a first section that describes the *MATLAB* commands that we will be using in the lab. Read through those quickly, so that you know what they are, and remember to refer back to this section as you work the lab for help with *MATLAB*. This lab also includes the important points we found about *MATLAB* in Lab 0. Be sure to review those as well.

Next, we give an overview of the model that we introduced in the Prelab, and introduce the question that you will be answering in your lab report. The remainder of each part of the lab are the exercises that constitute the work that you will need to complete the lab report. The actual lab report assignment is given at the end of the Part B document.

You will complete all of the work for this lab **in pairs**, with a partner. You will write your lab report together, and submit just one copy of that.

2. Matlab

First, recall the basic rules we learned about *MATLAB* in Lab 0, which are given in the boxed figure below.

The *MATLAB* commands we use in this lab are `ode45`, `plot`, and related commands to adjust and decorate the graphs we are plotting. These are (re)introduced below.

2.1. **ode45.**

The function ode45 generates an approximation to the solution to an initial value problem $y' = f(t, y)$, $y(t_0) = y_0$. The command is

```
>> [tsol,ysol] = ode45( func_handle, [tmin,tmax], initial_v );
```

for example, to solve $y' = 2y$, $y(0) = 1$, on $0 \leq t \leq 5$, we would use

```
>> [tsol,ysol] = ode45( @(t,y) 2*y, [0, 5], 1 );
```

Note that we could define the function handle and then plug it in to `ode45`:

```
>> f = @(t,y) 2*y;
>> [tsol,ysol] = ode45( f, [0, 5], 1);
```

The output from ode45 is a list of two arrays: the first gives the t values at which the numerical approximation was found, and the second the y values of the approximation. Thus, in the example here, `tsol(1)` is the first t value at which we have an approximation (which will be t_0) and `ysol(1)` the corresponding y value (y_0), etc.

Important MATLAB information

(1) *MATLAB* requires operators: `*` for multiplication, etc.
(2) In *MATLAB*, every variable is a vector (array), or matrix.
(3) You can get components of one of these vectors by indexing them with an integer index that indexes from one: `v(3)`, or `A(1,2)`.
(4) *MATLAB* provides a list of all defined variables in the current session in the *Workspace* window. The variable `ans` is used for any unnamed calculation. Note that it gets overwritten with the next unnamed calculation.
(5) Consistent with *MATLAB*'s world-view, functions and operators are vectorized: thus, if `t` is a vector of values, `sin(t)` returns a vector of the sine of those values. Similarly, operators such as `*` and `^` will perform *matrix multiplication and exponentiation*. To apply these element-by-element, which is what we will want to do most of the time in this course, prepend a period: thus `t.^2` is a vector with each element of the vector `t` squared.
(6) To pass a function to another function, use a function handle, for example, `@sin`. We can define an anonymous function as `@(t) t^2` (a function that squares every element of the input), or, if it is a function of two variables, `@(t,v) 8.730 - 0.721*v` (or similar).
(7) We can run any *MATLAB* script file by referring to it by name in our scriptfile, or by typing it at the command prompt (>>). We will do this in many labs to avoid having to retype commonly used functions many times.

2.2. **plot.** The function `plot` is

```
>> plot( t_values, y_values );
```

for example, to plot the numerical solution suggested above,

```
>> plot( tsol, ysol );
```

Recall that `plot` supports arguments that determine the appearance of the graph:

```
>> plot( tsol, ysol, '-b', 'LineWidth', 2 );
```

2.3. **axis.** The `axis` command allows you to set the *x* and *y* domains of a plot:

```
>> axis( [xmin xmax ymin ymax] );
```

for example,

```
>> axis( [0 50 0 12] );
```

2.4. **hold.** With `hold on`, retain all previous plots on the current axis, even as more are plotted. With `hold off`, replace the current plot with a newly generated one:

```
>> plot( t_values1, y_values1 );
>> hold on;
>> plot( t_values2, y_values2 );
>> hold off;
```

2.5. **title, xlabel, ylabel.** Set the title, x- and y-axis labels for an existing plot:

```
>> title( 'Solution of y'' = f(t,y)' );
>> xlabel( 't' );
>> ylabel( 'y' );
```

2.6. **legend.** Set the legend for the plot:

```
>> legend( 'label for 1st plotted function', 'label for 2nd'
);
```

Provide as many labels as there were data sets plotted in the graph.

3. Background

In this lab, we are studying the Gompertz equation, a first-order ordinary differential equation which models the growth of cancerous tumors,

$$\frac{dy}{dt} = ry\ \ln(K/y).$$

The constants r and K in this equation are positive. The function $y(t)$ gives the volume of the tumor at time t. The initial condition, $y(0) = y_0$, must be positive (that is, greater than zero), and it is reasonable to pick some arbitrary small value for that.

4. Lab Report

In your lab report, you will write a paper that looks at the behavior of solutions to the Gompertz equation, and what it predicts about the growth of a cancer tumor. While we are able to solve the Gompertz equation exactly, this is not the case for many nonlinear differential equations, and we therefore look at how we can approximate the Gompertz equation by expanding the nonlinearity in a truncated Taylor series, and you will look at what that analysis tells you about the expected behavior of the Gompertz equation, and when the resulting approximations are valid. As you work through the lab, you will want to think about how the exercises you are doing provide insight on these aspects of the equation. The details of the report are described at the end of Part B of the lab.

5. Part A Exercises

Exercise 1. Choose positive values r_1, r_2, K_1, K_2, and make a single plot of four curves, each a solution to the Gompertz equation with initial value $y(0) = 1$. Use `ode45` to find these solutions (see section 1 if you do not remember how to do this). There should be one for each combination of r and K. What are the equilibrium solutions in each case?

Note that in *MATLAB*, `log(x)` is the natural log function $\ln(x)$.

Your figure will be easiest to read if your choices for K_1 and K_2 are not too far apart. Notice what the effect of different r and K are on the solutions to the differential equation.

Exercise 2. Next, take $r = 0.1$ and $K = 10$ and plot solutions to the Gompertz equation, generated with `ode45`, for a number of different initial conditions: $y(0) = 0.1$, 1, 5, 8, 10, and 15. Does the qualitative behavior of the solutions change with different initial conditions? Do the curves look like a known function?

Exercise 3. Finally, in the prelab we found approximations to the Gompertz equation by expanding the log term, and the other term in y, in terms of $(y-K)$. Check that you both have the same approximations before proceeding. For the following, take $r = 0.1$ and $K = 10$.

Generate solutions, using `ode45`, for the approximations to the Gompertz equation found truncating the series expansion at $n = 1$, $n = 2$, $n = 3$, and $n = 4$. *(Note: you may get an error with one of these. If you do, try solving on a smaller range of* t *values.)* Plot the solutions, along with the `ode45` solution for the full Gompertz equation. What happens as n gets larger?

Note that $n = 2$ does not result in the same solution behavior as the other approximations why is that? Work out y' when $t = 0$ for the $n = 1$, $n = 2$, and $n = 3$ approximations. What does that tell you?