Show all relevant work for credit. You will be marked for your work and your answer as appropriate. Talking to other students about the problems is encouraged but you must submit your own work and identify who you worked with at the top of your assignment. 

ALL WORK MUST BE STAPLED. WRITE YOUR NAME AND STUDENT NUMBER AT THE TOP OF THE FIRST PAGE. MESSY WORK WILL NOT BE GRADED.

Problem 1: In each case, solve for \( y(t) \):

a. \( y' + 2y = 0 \) with \( y(0) = 1 \).

b. \( y' + 2y = -3e^t \) with \( y(0) = -2 \).

c. \( ty' - 2y = t^3 \sin t \) with \( y(\pi) = 0 \).

Problem 2: Use Matlab (dirfield) to generate a direction field for each equation from problem 1. Make plots on the range \( 0 < t < 4 \), \( -4 < y < 4 \). Superimpose plots (ezplot) of the analytic solutions corresponding to your answers from problem 1. Submit: the commands you used to achieve this, and your properly labelled plots.

To combine multiple plots together you can use the commands hold on and hold off. After obtaining the first plot type hold on, then all subsequent commands plot in the same window. After the last plot command type hold off.

Problem 3: (a) Use Matlab to generate the slope field of \( y' = y - \sin(t) + 1 \) on \( 0 < t < 2 \), \( -4 < y < 4 \). Obtain the solution curve for initial condition \( y(0) = -1 \) using ode45 and superimpose on the slope field. Submit: the commands you typed into Matlab to achieve this, and your properly labelled plot.

(b) Suppose that the initial condition is \( y(0) = \alpha \) and \( \lim_{t \to \infty} y(t) = -\infty \). What can you say about \( \alpha \)? To resolve this question you can make a guess based on the direction field plot, but you will need to solve the equation analytically to be confident of your answer.

Problem 4: Your lab partner leaves a drop of bleach on the lab bench, which takes the shape of a hemisphere. The drop initially has a radius of 1.6mm, and evaporates at a rate proportional to its surface area. After 10 minutes, the radius is 1.5mm. How long until the drop is gone?

Hint: you can solve this by generating a differential equation for the volume of the drop, \( dV/dt = \ldots \) or with a little more work, a differential equation for the radius of the drop, \( dr/dt = \ldots \). The latter equation is much easier to solve.

Problem 5: Find all values of \( r \) so that the function \( y = x^r \) solves the differential equation

\[
x^2 y'' - 3xy' - 32y = 0.
\]

Problem 6: In a murder investigation a corpse is found at exactly 8:00 p.m. The temperature of the body is immediately measured and is found to be 70°F (Fahrenheit). Two hours later, it is again measured and found to be 60°F. If the room temperature is 50°F, and we assume that Newton’s law of cooling applies,
when did the murder occur? (Assume that the temperature of the body at the time of the murder was 98.6°F).

Newton’s law of cooling states that the temperature $T$ of an object in a room at temperature $T_0$ follows the equation

$$\frac{dT}{dt} = k(T_0 - T).$$

**Problem 7:** In the previous problem, you probably concluded that the time of the murder was about 2.6 hours before the first temperature reading. However, someone points out your analysis is faulty because the room temperature in which the corpse was found was not constant but instead decreased exponentially according to the law $50e^{-0.05t}$, where $t$ is the time (in hours), where $t = 0$ corresponds to 8:00 p.m.

- (i) What is the differential equation that you must solve now?
- (ii) Solve the equation to find an analytic expression for the temperature of the body at any time $t$.
- (iii) Use Matlab (for example `ezplot`, see the end of this document) to make an accurate plot of the body temperature as a function of time. Estimate the time of the murder. Submit: your plot and the estimated time of death, as well as the commands you used.

**Problem 8:** Stephan’s law of radiative cooling states that the temperature $T$ of a material at time $t$ satisfies

$$T' = -k\left(T^4 - T_{env}^4\right)$$

where $k > 0$ is constant and $T_{env}$ is the constant environmental temperature. This law holds for bodies heated to a very high temperature that will emit blackbody radiation (i.e. molten iron).

- (i) By separating variables and factoring $T^4 - T_{env}^4 = (T^2 - T_{env}^2)(T^2 + T_{env}^2)$, show that the general solution to this equation is

$$\ln\left(\frac{T + T_{env}}{T - T_{env}}\right) + 2\tan^{-1}\left(\frac{T}{T_{env}}\right) = 4T_{env}^3kt + c$$

where $c$ is a constant.
- (ii) Let $T(0) = T_0 > T_{env}$. Calculate $\lim_{t \to \infty} T(t)$. 

2
Notes on Matlab/Octave usage for Homework 1

Some start-up material on Matlab syntax is available via the course home page (scroll down) or directly from:


If you have not used Matlab before, then this document would be a good place to start.
If you are using Octave: You should be aware that there are some small differences:

http://wiki.octave.org/FAQ#How_is_Octave_different_from_Matlab.3F

Using @-functions in Matlab/Octave

For this class we will need to define functions, for instance so that we can use them as parts of ODEs. You can define functions in Matlab/Octave using the @ syntax. For instance,

```matlab
>> f = @(x) exp(x)*sqrt(x+2) - 4
```
defines the function
\[ f(x) = (e^x \sqrt{x+2}) - 4. \]
To evaluate the function, you call as usual:

```matlab
>> f(0.3)
```
To graph the function you can use `ezplot` as follows (the function `plot` is for plotting vectors):

```matlab
>> ezplot(f,[0,10])
```
You can ignore complaints from matlab for now. You can also, for instance, find a zero of the function near an initial guess:

```matlab
>> fzero(f,0.5)
```

Plotting a direction field for a 1st order ODE

First download the file `dirfield.m` from the class website and put it in a folder that Matlab/Octave can find. You will need to make sure that the path contains this folder. This can be done from the “Set Path” button at the top of the command window, or from the command line: to view the current path, type `path`, add a new folder by syntax like `path(path,'c:/tools/goodstuff')` on Windows or `path(path,'/home/tools/goodstuff')` on Mac or Linux).

The file defines a command `dirfield` that you can call as with any other Matlab function. If you inspect the file you can see that it is mainly a friendly front-end to the `quiver` command which is a built in function for plotting vector fields.

Now to use `dirfield`, first define an @-function f of two variables t, y corresponding to the right hand side of the differential equation \[ y'(t) = f(t,y(t)). \] For example, for the differential equation \[ y'(t) = ty^2 \] define

```matlab
>> f = @(t,y) t*y^2
```
`dirfield` requires you to use this exact form, \( \Theta(t,y) \), even if \( t \) or \( y \) does not occur in your formula.

To plot the direction field for \( t \) going from \( t0 \) to \( t1 \) with a spacing of \( dt \) and \( y \) going from \( y0 \) to \( y1 \) with a spacing of \( dy \) use

```matlab
>> dirfield(f,t0:dt:t1,y0:dy:y1)
```
For example for \( t \) and \( y \) between -2 and 2 with a spacing of 0.2 type

```matlab
>> dirfield(f,-2:0.2:2,-2:0.2:2)
```
Basic numerical solution of a 1st order ODE

Define your ODE as $y' = f(t, y)$ and set up the function $f(t, y)$ as an @-function. Then use the command \texttt{ode45} to get Matlab to implement a 4th-order Runge Kutta method. This is a pretty reliable method for numerical integration of ODE. \texttt{ode45} is probably the most-used numerical ODE routine in the world. The syntax you need to create the solution to $y'(t) = f(t, y), y(a) = y_0$ in the interval $a \leq t \leq b$ is

\[
[tS,yS] = \texttt{ode45}(f,[a,b],y0)
\]

The output here is two vectors which in this example I have called \texttt{tS} and \texttt{yS}. Te vector \texttt{tS} contains a list of $t$-values and \texttt{yS} contains the values of $y(t)$ at those particular times. We can then plot the numerical solution (the pairs of points ($tS,yS$))by simply typing

\[
\texttt{plot(ts,ys)}
\]

which joins the points together with lines. To see the points themselves, you can type

\[
\texttt{plot(ts,ys,".")}
\]

Type \texttt{help plot} for more information on \texttt{plot} (or use Google). You can’t use \texttt{ezplot} for this purpose – that is meant for plotting functions rather than lists of points.

Obtaining numerical values of the solution

You can see by inspecting the time-point vector ($tS$ above) that \texttt{ode45} does not just use a uniform grid of time points. It can happen that you want to know the values of $y(t)$ at intermediate time points. To get those explicit values, specify the time points in a vector say \texttt{tV} and then apply \texttt{ode45} as follows:

\[
[tS,yS] = \texttt{ode45}(f,tV,y0)
\]

The first element of \texttt{tV} must be the start time and the last element must be the end time. The intermediate values are the values where you want to know $y$. For instance, if you wanted to know the solution every 0.5 time units on a time range from $-3$ to 7, you could write:

\[
tV = -3:0.5:7
\]

\[
[tS,yS] = \texttt{ode45}(f,tV,y0)
\]

Hint: put a semicolon on the end of any command to suppress output in the command window.