Differential Equations, Spring 2017
Computer Project 2, Due 11:30 am, Friday, March 10

The goal of this project is to implement the Euler Method and the Improved Euler Method, and to use these methods to numerically approximate the solution of a differential equation. Submit the following files on Canvas:

```
temp.m
EulerMethod.m
ImprovedEulerMethod.m
```

Make sure the files have exactly these names.

Here is a warm-up exercise to teach you about functions with multiple return values. Do not hand it in. Create a function called `test2.m` in your directory (I recommend creating a separate directory for each computer project). Enter the following code into the function:

```
function [a b]=test2()
    a=[5 6];
    b=10;
end
```

The function takes in 0 inputs and returns 2 outputs, `a` and `b`. You can run it from the command line by typing in

```
test2()
```

You can save its outputs to some variables by typing in

```
[x y]=test2();
```

This will store the value of the first output into `x` and the value of the second ouput into `y`. If you type in

```
z=test2();
```

only the first output will be stored into the variable `z`. You can create functions with an arbitrary number of outputs. For example, if the function began as

```
function [a b c]=test2()
```

then it would have 3 outputs.

Now on to the main project. The IVP we will study is

$$\begin{aligned} \frac{dy}{dx} &= .1(80 + 15\sin(2\pi x/24) - y) \\ y(0) &= 70. \end{aligned}$$

If the $15\sin(2\pi x/24)$ term were not present, you could think of this as being a usual Newton's Heating and Cooling ODE, with y the temperature and x the time (say in

hours). For instance, you could think of y as the temperature of a house (with no AC) on an 80° day. Since the temperature outside actually varies roughly periodically, we can make this a more realistic model by adding in the $15\sin(2\pi x/24)$ term. Notice that this term has period 24 hours. You can think of $80 + 15\sin(2\pi x/24)$ as being the ambient temperature on a summer day in Oklahoma. Thus the ODE is an improved version of Newton's Heating and Cooling ODE.

1. First we need to create the function $f(x, y) = .1(80+15\sin(2\pi x/24)-y)$ as a Matlab function. We will call the function `temp`. Here is the outline. You need to fill in the ...

   ```
   function val=temp(x,y)
       val=...;
   end
   ```

   Test that it works by entering a command such as

   ```
   temp(2,3)
   ```

2. Now we will create the program to implement the Euler Method. Create a function called `EulerMethod`. Here is the outline of the function code.

   ```
   function [xvals yvals]=EulerMethod(f,xmin,xmax,steps,y0)
   % calculate the step size
   h=(xmax-xmin)/steps;
   % now set up a list of x values
   % need the list to have steps+1 numbers in it
   % because Matlab counts the initial x as 1 (instead of 0)
   xvals=linspace(xmin,xmax,steps+1);
   x=xmin;
   % initialize y and yvals
   y=y0;
   yvals=[y];
   % now a for loop to build the yvals
   % inside the loop, x and y will be variables with values the current (ith) point
   % you need to find the next (i+1st) y value
   for i=1:steps
       % x and y already have values stored in them
       % use these values to find slope
       slope=f(x,y);

       ...

       y=...;  %this should be the new y value
       yvals=[yvals y]; %add new y value to the list of previous yvals
   ```

```
    x=xvals(i+1); %update x to the new x value
end
end
```

You need to fill in the middle. The function takes in 5 inputs. The first input is the name of the function for the ODE. For this project it will be the function `temp` we created in Step 1. (The syntax for giving this function as an argument is `@temp`, see the test code at the end of this step.) The second input is the initial x value, the third input is the final x value, the fourth input is the number of steps to use, and the final input is the initial y value.

The function returns 2 outputs. The first output is a list of the x values, and the second output is a list of the corresponding y values. These values are the numerical approximate solution of the differential equation.

Lines that start with `%` are comment lines. They have no effect on the code and I put them in there to tell you a little bit about what the code is doing. The only thing you need to do to complete the program is to remove the ... and put in the correct formula for `y` in the for loop.

Now here is some code you can run to test it.

```
[x y]=EulerMethod(@temp,0,4*24,1000,70);
plot(x,y)
```

This will approximate the solution over the interval $[0, 4 * 24]$ (4 days) using 1000 steps.

3. Now create a file called `ImprovedEulerMethod.m` to implement the Improved Euler Method. You can start by reusing the outline of the previous code. Here it is to get you started:

```
function [xvals yvals]=ImprovedEulerMethod(f,xmin,xmax,steps,y0)
% calculate the step size
h=(xmax-xmin)/steps;
% now set up a list of x values
% need the list to have steps+1 numbers in it
% because the initial x is counted as 1
xvals=linspace(xmin,xmax,steps+1);
x=xmin;
% initialize y and yvals
y=y0;
yvals=[y];
% now a for loop to build the yvals
for i=1:steps
    % x and y already have values stored in them
    % use these values to find slope
```

```
    slope=f(x,y);

    ...

    y=...;
    yvals=[yvals y];
    x=xvals(i+1);
end
end
```

You can test that it works by running the following test code.

```
[x y]=ImprovedEulerMethod(@temp,0,4*24,1000,70);
plot(x,y)
```

4. That is it for things you have to turn in. Here are some commands you can run to see the effect of increasing the step size, and also to compare the two methods. (You probably will want to create a script to hold these commands.) Also, the last commands plot the curve $y = 80+15\sin(2\pi x/24)$, which is the ambient temperature (note: this is not the exact solution of the ODE). If the output is too complicated to understand, comment out some of the plot commands and then run it again to get a simpler output. You should be able to see that the Improved Euler Method is a lot better.

   It is interesting to compare the ambient temperature to the temperature of the house (comment out all but the last two plot commands to better see this). Notice that the exact solution has a different amplitude than the ambient temperature. Both curves have a period of 24 hours, but the exact solution has a so-called phase shift from the ambient temperature. This just means that the max points of the two curves don't occur at the same times. In other words, the peak temperature in the house does not occur at the hottest time of the day.

```
hold on
[x y]=EulerMethod(@temp,0,4*24,5,70);
plot(x,y)
[x y]=EulerMethod(@temp,0,4*24,10,70);
plot(x,y)
[x y]=EulerMethod(@temp,0,4*24,20,70);
plot(x,y)
[x y]=ImprovedEulerMethod(@temp,0,4*24,20,70);
plot(x,y)
[x y]=ImprovedEulerMethod(@temp,0,4*24,20,70);
plot(x,y)
[x y]=ImprovedEulerMethod(@temp,0,4*24,20,70);
plot(x,y)
%% the next one is pretty much the exact solution
```

```
[x y]=ImprovedEulerMethod(@temp,0,4*24,10000,70);
plot(x,y)
%% next is code to plot the ambient temp
xvals=linspace(0,4*24,100);
yvals=[];
for x=xvals
    y=80+15*sin(2*pi*x/24);
    yvals=[yvals y];
end
plot(xvals,yvals);
```