APPM 2460
NUMERICALLY SOLVING ODES

1. Introduction

Today we'll review numerically solving ordinary differential equations. This is one of the key uses of Matlab we will see this semester.

What does it mean to "numerically" solve an ODE? Suppose we have the initial value problem

$$y' = t + y \tag{1}$$

Take a moment to solve this ode. The equation isn't separable, so you'll likely have to use a different method. Often times we can't solve an ODE by hand which means that we need to find an approximate solution.

> Numerically solving an ODE gives us an **approximate** solution.
> It is most helpful when we cannot find an **exact** solution.

You will also see the phrase "analytical solution" used in place of "exact solution." **When you solve an ODE using a pen and paper (e.g. by separation of variables, integrating factor, or any other method) you are finding the analytical solution.** For this homework assignment, we will be solving (1) numerically. To ensure that we have the correct numerical solution, we can compare to the analytical solution.

There are myriad methods that can be used to numerically (read: approximately) solve an ODE. Matlab has many of these methods built in, and the development and analysis of such methods is an active area of research here in the applied math department. We will only discuss the most basic method, known as Euler's method today.

2. Euler's Method

Today we will build our own functions so we can solve difficult problems like (1). Typically when we build approximate solvers for ODEs, we should test our code on a problem that we can actually solve–we will implement this process today.

Recall that Euler's method to approximate a solution to the initial value problem

$$y'(t) = f(t, y), \quad y(0) = y0, \tag{2}$$

is

$$y_{n+1} = y_n + hf(t_n, y_n), \quad y_0 = y0. \tag{3}$$

We've had some practice doing this on our 2360 homework this semester, and I'm sure that you've found it to be quite tedious to go through this calculation. That's why we want to use Matlab to do all the tedious calculations for us.

3. Euler's Method in Matlab

If we want to implement Euler's method we first rewrite the differential equation as

$$y'(t) = f(t, y). \tag{4}$$

The following outlines one method we can use to implement Euler's method in Matlab. First we write a **function** to define the right hand side of the ode.

```
function yp = yprime(t,y)
%% define f to be the right hand side of the
%% ode that takes input values (t,y) and returns y'

yp = f(t,y) %% f(t,y) is the right hand side of the ode

end
```

Now we can write a **script** that can actually preform Euler's method.

```
%% This script will use Euler's method to compute the solution of a
   differential equation y' = f(t,y) with initial conditions y(t0) = y0.
    The approximate solution is a vector "y" with values corresponding
   to a vector of times "t"

h = ; %specify the step size in Euler's method
t = t0:h:t_end; %specify the time interval
y(1) = y0; %set initial condition

for n = 1:(length(t)-1)
        y_{n+1}  = y_{n} + h*yprime(t_n,y_n) %Do Euler's method
end
```

Note the above code is not consistent with Matlab syntax. Subscripts can be interpreted as the index within a vector in order to translate mathematical expression to Matlab code.

4. Homework

Use Euler's method to approximate the solution to the initial value problem

$$y'(t) = t + y, \quad y(0) = 1.$$

To do this pick $t \in [0, 10]$ and $h = 1$, $h = 0.5$ and $h = 0.1$. Compare the approximate solution to the true solution by plotting both on the same axes.

Protip: If you carefully write this code you can perform only slight modifications of it for project 1 in APPM 2360.