# LAB 2: Euler's Method

In this lab you'll develop your knowledge of MATLAB/Octave by implementing Euler's Method for IVPs, and determining, experimentally, its order of accuracy.

At the end of the class, upload your code to Blackboard (go to "Assignments and Labs" and then "Lab 2"). In Lab 3, you'll implement higher-order schemes.

## 1 Four ways to define a vector

The most fundamental object in MATLAB is a matrix. The the simplest (nontriveal) example of a matrix is a vector. So we need to know how to define vectors. Here are several ways we could define the vector

$$x = (0, .2, .4, \dots, 1.8, 2.0). \tag{1}$$

```
x = 0:0.2:2 % From 0 to  2 in steps of 0.2
x = linspace(0, 2, 11); % 11 equally spaced
              % points with x(1)=0, x(11)=2.
x = [0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, ...
  1.6, 1.8, 2.0]; % Define points individually
```

The last way is rather tedious, but this one is even more so:

```
x(1)=0.0; x(2)=0.2, x(3)=0.4; x(4)=0.6; ...
```

In Section 3 we'll see how to do this using a `for`-loop.

## 2 Script files

MATLAB and Octave are *interpretative* environments: if you type a (correct) line of code, and hit return, it will execute it immediately. For example: try `>> exp(1)` to get a decimal approximation of $e$.

However, we usually want to string together a collection of MATLAB operations and run them repeatedly. To do that, it is best to store these commands in a *script* file. This is done be making a file called, for example, *Lab2.m* and placing in it a series of MATLAB commands. A script file is run from the MATLAB command window by typing the name of the script, e.g., `>> Lab2`

Try putting some of the above commands for defining a vector into a script file and run it.

## 3 `for`–loops

When we want to run a particular set of operations a fixed number of times we use a `for`–loop.

It works by iterating over a vector; at each iteration the *iterand* takes each value stored in the vector in turn. For example, here is another way to define the vector in (1):

```
for i=0:10 % 0:10 is the vector [0,1,...,10]
  x(i+1) = i*0.2;
end
```

## 4 Functions

In MATLAB we can define a function in a way that is quite similar to the mathematical definition of a function. The syntax is `>> Name = @(Var)(Formula);` Examples:

```
f = @(x)(exp(x) - 2*x -1);
g = @(x)(log(2*x +1));
```

Now, for example, if we call $g(1)$, it evaluates as $\log(3) = 1.098612288668110$. Furthermore, if `x` is a vector, so too is `g(x)`.

A more interesting example would be to try

```
>> xk = 1;
```

and then repeat the line

```
>> xk = g(xk)
```

Try this and observe that the values of `xk` seem to be converging. This is because we are using *Fixed Point Iteration.*

Later we'll need to know how to define functions of two variables. This can be done as:
`F = @(y,t)(y./(1 + t.^2));`

## 5 Plotting functions

MATLAB has two ways to plot functions. The easiest way is to use a function called `ezplot`:

```
 ezplot(f, [0, 2]);
```

plots the function $f(x)$ for $0 \le x \le 2$. A more flexible approach is to use the `plot` function, which can plot one vector as a function of another. Try these examples below, making sure you first have defined the vector $x$ and the functions $f$ and $g$:

```
figure(1);
plot(x,f(x));
figure(2);
plot(x,f(x), x, g(x),  '--', x,x, '-.');
```

Can you work out what the syntax `'--'` and `'-.'` does? If not, ask a tutor. Also try

```
plot(x, f(x), 'g-o', x, g(x),  'r--x', ...
   x,x, '-.');
```

## 6 How to learn more

These notes contain just enough information to get started. There are many good books available, electronically, through the University library, e.g., e "Learning MATLAB" by Tobin Driscoll.

## 7 Initial Value Problems

The particular example of an IVP that we'll look at in this lab is: *estimate $y(4)$ given that*

$$y'(t) = y/(1+t^2), \text{ for } t > 0, \qquad \text{and } y(0) = 1. \qquad (2)$$

The true solution to this is $y(t) = e^{\arctan(t)}$.

## 8 Euler's Method

**Euler's Method** is

- Choose $n$, the number of points at which you will estimate $y(t)$. Let $h = (t_n - t_0)/n$, and $t_i = t_0 + ih$.
- For $i = 0, 1, 2, \dots, n-1$ set $y_{i+1} = y_i + hf(t_i, y_i)$.

Then $y(t_n) \approx y_n$. As shown in Section 2.3 of MA385, the global error for Euler's method can be bounded:

$$|\mathcal{E}_n| := |y(T) - y_n| \le Kh,$$

for some constant $K$ that does not depend on $h$ (or $n$). That is, if $h$ is halved (i.e., $n$ is doubled), the error is halved as well.

Download the MATLAB script file `Euler.m`. It can be run in MATLAB simply by typing `>> Euler` It implements Euler's method for $n = 1$. Read the file carefully and make sure you understand it.

The program computes a vector `y` that contains the estimates for $y$ at the time-values specified in the vector `t`. However, MATLAB indexes all vectors from 1, and not 0. So `t(1)` $= t_0$, `t(2)` $= t_1$, ..., `t(n+1)` $= t_n$.

Use the program to compute $\mathcal{E}_n$ for $n$ = 2, 4, 8, ..., 512, and tabulate the results. We want to use this table to verify that Euler's Method is 1st-order accurate. That is:

$$|\mathcal{E}_n| \le Kh^\rho \qquad \text{with} \qquad \rho = 1.$$

A computational technique that verifies the order of the method is to estimate $\rho$, for a given $n$, by

$$\rho_n \approx \log_2\Big(\frac{|\mathcal{E}_{n/2}|}{|\mathcal{E}_n|}\Big). \qquad (3)$$

| $n$ | $y_n$ | $\mathcal{E}_n$ | $\rho_n$ |
|---|---|---|---|
| 2 | 4.2 | $4.347 \times 10^{-1}$ | — |
| 4 | 3.96 | $1.947 \times 10^{-1}$ | |
| 8 | | | |
| 16 | | | |
| 32 | | | |
| 64 | | | |
| 128 | | | |
| 256 | | | |

## 9 Exercise

Modify the code provided so that it uses a `for`-loop to compute the errors for $n = 2, 4, 8, \dots$. The code should output $\mathcal{E}_n$ for each $n$, and also estimate the corresponding $\rho_n$ for $n = 4, 8, \dots$.

Finally, the program should produce a "log-log" plot of both $\mathcal{E}_n$ and $n.^{-1}$ as functions of $n$, which also demonstrates the convergence of the method.

**Upload your code to the `Assignments -- Lab 2` section of Blackboard**. Take care to include useful comments, and, especially, include your Name, ID Number and Email Address.

**DEADLINE: 5pm, Friday 1 Nov, 2019**

## 10 More MATLAB: formatted output

The `Euler.m` script file uses the `fprintf` function to display messages and values. Its basic syntax is:

```
fprintf('Here is a message\n');
```

where the `\n` indicates a new line.

Using `fprintf` to display the contents of a variable is a little more involved, and depends on *how* we want the data displayed. For example:

- To display an integer:
  ```
  fprintf('Using n=%d steps\n', n);
  ```
- To display an integer, padded to a maximum of 8 spaces:
  ```
  fprintf('Using n=%8d steps\n', n);
  ```
- To display a floating point number:
  ```
  fprintf('y(n)=%f\n', Y(n+1));
  ```
- To display in exponential notation:
  ```
  fprintf('Error=%e\n', Error);
  ```
- To display 3 decimal places:
  ```
  fprintf('y(n)=%.3f\n', Y(n+1));
  fprintf('Error=%.3e\n', Error);
  ```

Use these to improve the formatting of the output from your `Euler.m` script so that the output is nicely tabulated. To get more information on `fprintf`, type

```
>> doc fprintf
```

or ask one of the tutors.