Lab 2: Euler's Method and RC Circuits

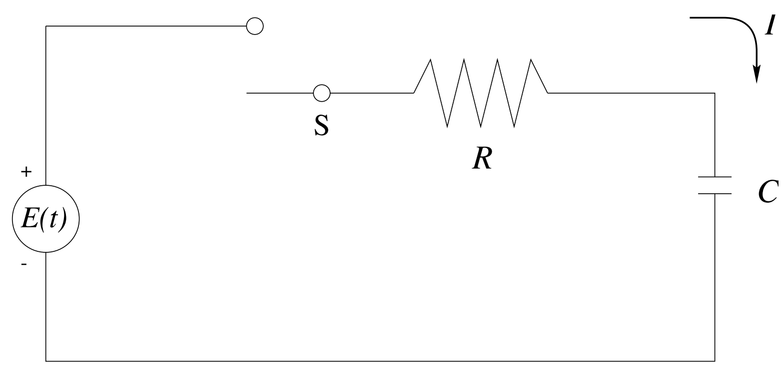
Goals

In this lab you will implement Euler's method to approximate measurements of the charge on a capacitor in a basic RC circuit. You will learn how to write `.m` files for Matlab and how to program Euler's method; then you will investigate some of the limitations of the method.

Application: a basic RC circuit

The state of an electrical circuit consisting of resistors, capacitors, and an applied voltage can be described by differential equations.

Consider the following circuit



with resistance R Ohms (Ω), capacitance C Farads (F), and applied voltage $E(t)$ Volts (V). The charge on the top plate of the capacitor at time t is $Q(t)$ Coulombs (C), and the current through the resistor is $I(t)$ Amperes (A). The resistor has a voltage drop of RI, and the capacitor has a voltage drop of Q/C. When switch S is closed at time $t = 0$, the sum of the voltage across the resistor and the capacitor must equal the applied voltage. This gives us the equation

$$RI + \frac{1}{C}Q = E(t)\,.$$

The current in the circuit is the rate of change of the amount of charge on the capacitor. So using the relationship $dQ/dt = I$, this becomes a first order differential equation for $Q(t)$:

$$R\frac{dQ}{dt} + \frac{1}{C}Q = E(t)\,.$$

The initial condition for this equation is $Q_0 = Q(0)$, the initial amount of charge on the capacitor. (Q_0 could be set by imposing a voltage $V_0 = Q_0/C$ across the capacitor before inserting it into the circuit.) In this lab, we will use Euler's method to numerically solve this differential equation for two different applied voltages: a constant voltage E_0, and then an AC voltage $E(t) = 117\sin(120\pi t)$ which corresponds to the voltage out of a standard wall socket.

Prelab assignment

Before arriving in lab, answer the following questions. You will need your answers in lab to work the problems, and your recitation instructor may check that you have brought them. These problems are to be handed in as part of your lab report.

1. (a) Verify that the function

$$Q_1(t) = E_0 C\left(1 - e^{-t/(RC)}\right) \tag{1}$$

is a solution to the initial value problem

$$\frac{dQ_1}{dt} = -\frac{1}{RC}Q_1 + \frac{1}{R}E_0\,, \qquad Q_1(0) = 0\,, \tag{2}$$

where R, C, and E_0 are constants. That is, by plugging $t = 0$ into the formula (1) show that the initial condition is satisfied, and then by differentiating the formula (1) and comparing with the right-hand side of the differential equation show that $Q_1(t)$ satisfies the differential equation. (In other words, do not try to find the solution of the initial-value problem, but rather just check that the given function solves the problem.) Then use the exact solution formula (1) with $R = 20000\Omega$, $C = .00001$F, and $E_0 = 117$V to complete the column labelled "Exact y" on Table 2 on the last page of the lab, for use in Lab problem 1.

(b) Verify that the function

$$Q_2(t) = \frac{E_0}{R\left(\gamma^2 + (120\pi)^2\right)}\left[120\pi\left(e^{-\gamma t} - \cos(120\pi t)\right) + \gamma\sin(120\pi t)\right] \tag{3}$$

satisfies the initial-value problem

$$\frac{dQ_2}{dt} = -\frac{1}{RC}Q_2 + \frac{1}{R}E_0\sin(120\pi t)\,, \qquad Q_2(0) = 0\,, \tag{4}$$

where the constant $\gamma = 1/(RC)$ has units of Hz=sec^{-1}. What do these initial conditions represent for the system at the time the switch is closed?

2. Suppose you implement Euler's method using Matlab, using step size h, and create a vector `t` of time steps from $t = 0$ to $t = 1$. Often we refer to the first entry as $t_0 = 0$, the next as t_1 and the final entry will be $t_N = 1$ where $Nh = 1$. Matlab does not enumerate these entries in the same way. The first element of the vector is always `t(1)`. In this case, we will have `t(1)=0`, and `t(N+1)=1`. Find the Matlab indices `n` so that `t(n)=0`, `t(n)=.5`, `t(n)=.86`, and `t(n)=1` if you used

(a) $N = 10$ (Note: you cannot get `t(n)=.86` in this case.)

(b) $N = 100$

(c) $N = 500$

(d) $N = 1000$

Record these values of `n` in Table 1 below.

N	n so that `t(n)=0`	n so that `t(n)=.5`	n so that `t(n)=.86`	n so that `t(n)=1`
10				
100				
500				
1000				

Table 1: Matlab Indices for Time Vector

In the lab

Working with Matlab, we can write simple programs that implement numerical methods like Euler's method and compare the results obtained for several step sizes h with exact solutions (when these are available).

Creating a (new) file

Launch Matlab. Use the buttons to the right of the current directory to select the folder `~/Desktop/IFS` as your working directory. (The directory `~/Desktop/IFS` is in fact a link to your home directory that you access from any of the public university computers that you log into using your uniqname; you may wish to organize your home directory by first making a subdirectory called `Math216`, in which case you should choose `~/Desktop/IFS/Math216` as your working directory when you are using the computers in the labs in East Hall.) Then go to the File menu and use the mouse button to select New and then M-file; a new window called untitled will open. Choose Save As... from the File menu and a dialog box will open. In the box labelled Save that appears there is a default entry `untitled.m`. Replace that with `EULER.m` and click the Save button. The editor window should now be titled EULER.m. You have created an (empty) file called `EULER.m`. (Matlab file names all end in `.m`). It is important here that `EULER` is capitalized.

The contents of the `EULER.m` file

This program will implement Euler's method to solve the initial-value problem

$$\frac{dy}{dt} = f(t, y)\,, \qquad y(a) = y_0\,. \tag{5}$$

In this application, the function $y(t)$ stands for $Q(t)$, the charge on the capacitor. In the program below, everything following a % is a *comment*. Comments give you information about the program, but are not evaluated by the computer. You may choose whether or not to type these comments into your program, but if you include the comments in your file you must include the %, or the computer will try to read them. Type the following program into the file `EULER.m` you have just created:

```
clear t                             % Clears old time steps and
clear y                             % y values from previous runs
a=0;                                % Initial time
b=1;                                % Final time
N=100;                              % Number of time steps
y0=0;                               % Initial value y(a)
h=(b-a)/N;                          % Time step
t(1)=a;
y(1)=y0;
for n=1:N                           % For loop, sets next t,y values
  t(n+1)=t(n)+h;
  y(n+1)=y(n)+h*f(t(n),y(n));  % Calls the function f(t,y)=dy/dt
end
plot(t,y)
title(['Euler Method using N=',num2str(N),' steps, by MYNAME'])
                                    % Include your own name
```

Substitute your own name in place of `MYNAME` so it will appear on the plots you create. When you are finished typing in the program statements, save your work by choosing Save from the File menu, or by clicking on the floppy disk icon on the EULER.m editor window.

The right-hand side of the differential equation and the file `f.m`

Since the program `EULER.m` refers to the function `f(t,y)`, we must create a second `.m` file called `f.m` to define the right-hand side of the differential equation for Matlab. Go to the File menu and select New then M-file. A new window called untitled will open. Choose Save As... from the File menu and a dialog box will open with a default entry `untitled.m`. Replace that with `f.m` and click Save as you did before. You should then type into the file `f.m` the following commands:

```
function f=f(t,y)
R=20000;            % Resistance
C=.00001;           % Capacitance
E0=117;             % Constant Voltage
f=-y/(R*C)+E0/R;    % Defines the function f
```

Be sure to save your work when you have finished typing.

The file f.m contains the function $f(t, y)$ for the general differential equation of the initial-value problem (5) above; the particular form of $f(t, y)$ appearing on the last line corresponds to the specific initial-value problem (2). To solve a different differential equation with EULER.m or another solver, you need only change this file. The other lines in f.m serve to define specific values for the constants; here we are using $R = 20\text{k}\Omega$, $C = 10\mu\text{F}$, and $E_0 = 117\text{V}$.

The exact solution of the initial-value problem and the file yE.m

Since for this problem we also know a formula for the exact solution to the initial-value problem, we can write a short Matlab program to evaluate this formula so we can compare with the results of using Euler's method for various step sizes h. To tell Matlab about the exact solution formula, you should create another new file called yE.m. The file yE.m will contain the exact solution $Q_1(t)$ of the initial-value problem (2) corresponding to the right-hand side function $f(t, y)$ defined in the file f.m. If you solve a different differential equation with EULER.m or one of the other numerical methods we will study in Lab 3, and you wish to compare with an analytical expression for the exact solution, you should modify the file yE.m as well as f.m. Type the following commands into yE.m, saving your work when you have finished:

```
function yE=yE(t)
R=20000;                    % Resistance
C=.00001;                   % Capacitance
E0=117;                     % Constant Voltage
gam=1/(R*C);
yE=E0*C*(1-exp(-gam*t));    % Exact solution yE
```

Running your program

So far you have written a program and saved it. Now you want to use the program. In the Command Window type EULER. The plot of the solution curve should then appear in a new window.

Here is a summary of the programming we have just done. To solve an initial-value problem for a differential equation you need a solver. In this case, EULER is your solver. Then you must tell your solver what it is supposed to solve; that is your f.m file defining the right-hand side of the differential equation. Finally, the solver must be told where to start, so you must specify initial conditions. In this case the initial conditions are entered directly into the solver with the lines a=0 and y0=0. You 'fine tune' your solver by changing N. The stopping time is also entered in the solver as b=1.

Lab problems

1. Solve initial-value problem (2) using `EULER.m` with $N = 10$.

   (a) Print your results as follows: After the plot has appeared, go to the File menu on the figure window and choose Print...; a dialog box will appear. If you click OK in the dialog box the plot will print on the printer in your lab.

   By the way, you can use the Edit menu on the figure window to change the size of your plot, add text, labels, legends, titles, etc.

   (b) Using your indices from Prelab problem 2 and your approximate solution vector `y` calculated by `EULER.m`, complete the portion of Table 2 corresponding to $N = 10$. To view the `n`th component of the approximate solution vector `y`, just type `y(n)` in the Matlab command window and press return. Similarly, to view the corresponding exact solution value, type `yE(t(n))` and press return. Make sure you have enough significant digits to compare the approximate and exact answers. You can see more significant digits in Matlab by typing `format long` in the Matlab command window and then hitting return.

2. Repeat Problem 1 again using $N = 100$ and $N = 1000$. Use this data to complete the rest of Table 2.

3. What guesses can you make about the error at a given time as N increases?

4. Graphically investigate the error in solving the initial-value problem (4). To solve this problem using `EULER.m`, you will have to change your file `f.m`, replacing the line

   ```
   f=-y/(R*C)+E0/R;
   ```

   with the line

   ```
   f=-y/(R*C)+(E0/R)*sin(120*pi*t);
   ```

   To compute the exact solution $Q_2(t)$ for the initial-value problem (4), you will need to modify the file `yE.m`, replacing the line

   ```
   yE=E0*C*(1-exp(-gam*t);
   ```

   with the lines

   ```
   omg=120*pi;
   A=E0*omg/(R*(gam^2+omg^2));
   B=E0*gam/(R*(gam^2+omg^2));
   yE=A*(exp(-gam*t)-cos(omg*t))+B*sin(omg*t);
   ```

   You can move among the various files in the editor window by clicking on the tabs at the bottom of the window.

   Important note: before modifying the files, save your original files `f.m` and `yE.m`, for instance by renaming them as `f1.m` and `yE1.m` with the help of the Save As... command from the File menu; also, save copies of these modified files as `f2.m` and `yE2.m`, as you will be able to reuse them in Lab 3. Make sure your working directory is (a subdirectory of) `~/Desktop/IFS` before saving to make sure that you will be able to access these files later when you are working on Lab 3.

To plot the approximate solution and the exact one on the same set of axes, replace the line `plot(t,y)` near the end of the file `EULER.m` with the lines

```
yexact=yE(t);
plot(t,y,':',t,yexact,'-')
legend('Approximate','Exact')
```

This defines a vector corresponding to the exact solution. Sample values of the exact solution are plotted connected by a solid line, and the Euler's method approximation is plotted on the same axes as a dotted line. The `legend` command puts a helpful box in the corner of the plot to help you identify which graph corresponds to which function. You can change the location of the legend on the graph by clicking on it or by using the Edit menu on the figure.

5. Run `EULER` using $N = 50, N = 100, N = 200, N = 500, N = 1000$ and $N = 2000$. Why does the exact solution look different on the first few plots? How large must N be for the approximate solution to qualitatively match the exact one? We say that two graphs agree qualitatively if they have roughly the same shape, for example, they might both be increasing or have two maxima or be periodic with approximately the same period. Print out a plot where the exact solution appears to have a lower frequency than 60 Hz, one where the approximate solution appears qualitatively correct, and one where the approximate solution appears quantitatively correct. (See if you can print out clearer plots with fewer oscillations by zooming by typing `axis([tmin,tmax,ymin,ymax])` with appropriate minimum and maximum values in the command window, or by changing the final time `b`.)

6. Many different situations can give rise to error in numerical approximations. One such situation, *undersampling*, comes in to play in this last problem. Undersampling occurs when the data points are spaced too far apart to capture all the behavior of the equation. You may know this phenomenon as "aliasing", when an insufficiently sampled high frequency signal appears to be a lower frequency signal. Even if you have calculated the exact solution correctly its graph may not be accurate. Imagine plotting a sine wave with frequency 120 cycles per second by sampling the function 120 times a second; the sample points will suggest a constant function instead of an oscillatory function. Did you see this problem when you used $h = 0.01$? What about $h = 0.001$? (Lab 3 will develop more numerical methods that are potentially more accurate.)

Note: your lab report for Lab 2 should consist of your solutions to the prelab problems, your solutions to lab problems 1–6, and a brief, original, conclusions paragraph summarizing what you have learned.

Time	Index	Exact y	Approximate y	Error	Percent Error
t	`n`	`yE(t)`	`y(n)`	|`y(n)`−`yE(t)`|	$\frac{\texttt{y(n)} - \texttt{yE(t)}}{\texttt{yE(t)}} \times 100$
$N = 10$					
.5					
1					
$N = 100$					
.5					
.86					
1					
$N = 1000$					
.5					
.86					
1					

Table 2: Approximate Solutions to Equation (2)