# Math242: Engineering Mathematics II

# MATLAB Homework Assignment

---

## Instructions:

- Prepare a pdf report that includes your answers/plots.
- Do **NOT** include your MATLAB code in the report.
- Use a single .m file for all MATLAB code you write.
- The code file should **NOT** return an error during runtime.
- Code should be commented on and code for different HW questions should be clearly separated.
- Compress the .m and .pdf files into a single .zip file and upload the zip file on Moodle.

---

## 1 Solving Ordinary Differential Equations

In this section, we will show how to solve differential equations with MATLAB by using the **dsolve** command to produce a general solution to the differential equation, or a specific solution to an associated initial value problem.

## 1.1 MATLAB Tutorial

Consider the following first-order linear ODE:

To solve this equation, first, represent *y* by using syms to create the symbolic function *y*(*t*).

```
>> syms y(t);
```

Define the equation using == operand and represent differentiation using the diff function.

```
>> ode = diff(y,t) == 5*t*y;
```

Solve the equation using dsolve and obtain *y*(*t*).

```
>> ySol = dsolve(ode)
ySol = C1*exp(5*t^2/2)
```

Now consider the following second-order ODE with initial conditions:

$$\frac{d^2y}{dx^2} = k\cos 2x - y \tag{2}$$

$$y(0) = 1, \tag{3}$$

$$y'(0) = 0. \tag{4}$$

Define the equation and conditions. The second initial condition involves the first derivative of *y*. Represent the derivative by creating the symbolic function Dy = diff(y) and then define the condition using Dy(0)==0.

```
>> syms y(x) k;
>> Dy = diff(y);
>> ode = diff(y,x,2) == k*cos(2*x)-y;
>> cond1 = y(0) == 1; cond2 = Dy(0) == 0;
>> conds = [cond1 cond2];
```

Solve ode for *y*. Simplify the solution using the simplify function.

```
>> ySol(x) = dsolve(ode,conds);
>> ySol = simplify(ySol)
ySol(x) = k/3 + cos(x) - (2*k*cos(x)^2)/3 + (k*cos(x))/3
```

Now that we obtained *y*(*x*), we can plot it over *x* using the ezplot function. ezplot(f(x)) plots the expression *f*(*x*) over the default domain $-2\pi < x < 2\pi$.

```
>> ezplot(ySol)
>> grid on
```

Figure 1 shows the resulting plot. We can use subs function to substitute a desired value of *k* into the solution *y*(*x*). Figure 2 plots *y*(*x*) for three different values of *k* = −10, 0, 10.

```
>>  subplot(1,3,1)
>>   ezplot(subs(ySol,k,-10))
>> grid  on
>>  subplot(1,3,2)
>>   ezplot(subs(ySol,k,0))
>> grid  on
>>  subplot(1,3,3)
>>   ezplot(subs(ySol,k,10))
>> grid  on
```

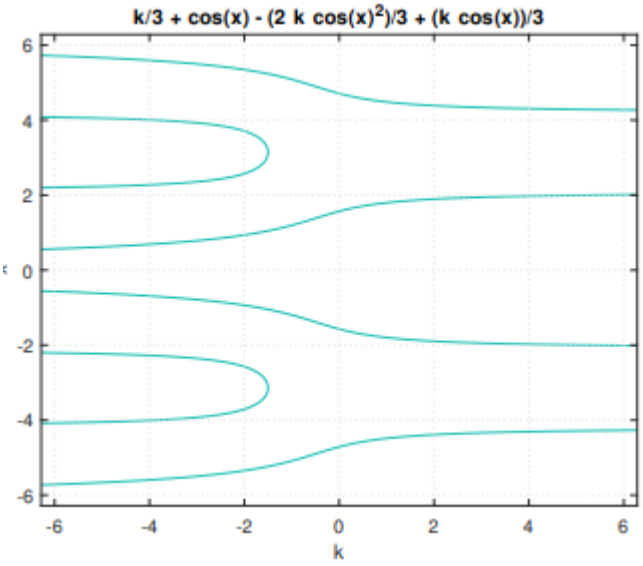



Figure 1: $y(x)$ for $x = [-2\pi, 2\pi]$ and $k = [-2\pi, 2\pi]$.

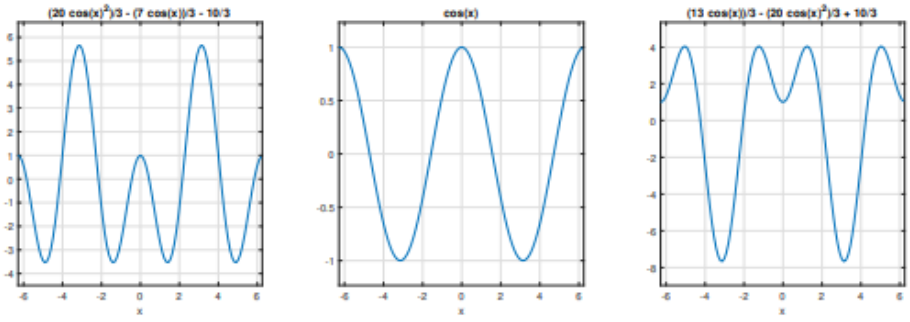



Figure 2: $y(x)$ for $x = [-2\pi, 2\pi]$ and $k = -10, 0, 10$.

## 1.2 Question 1

Solve the following initial value problems using Matlab, and plot the solution of the equations respectively,

a) $y' = \frac{3}{t}y + t^2e^t + \sin(t)\,, \quad 0 \le t \le 5, \quad y(1) = 0$

b) $y' = t^{-2}(\sin(2t) - 2ty), \quad 1 \le t \le 10, \quad y(1) = 2$

# 2 Numerical Methods

We have seen that many differential equations can be solved explicitly in terms of the elementary functions of calculus using dsolve. But many other differential equations cannot be solved explicitly in terms of the functions of calculus. In these cases, we use numerical methods to obtain information about the solution without the use of a solution formula.

## 2.1 Euler's Method

In mathematics and computational science, Euler's method is a numerical procedure for finding the approximate solution to an ordinary differential equations with a given initial value. There are many different methods that can be used to approximate solutions of a differential equation. Euler's Method is one of the oldest and easiest ones.
Let's start with a simple first order differential equation,

$$\frac{dy}{dx} = f(x.y) \qquad y(x_0) = y_0 \qquad (6)$$

where $f(x, y)$ is a known function and has known initial conditions. The main purpose is to approximate the solution near $x = x_0$. We know the value of the solution at $x_0$ and also the value of the derivative at $x = x_0$ by simply plugging the initial conditions into $f(x, t)$

$$\frac{dy}{dx} = f(xo, yo) \qquad (7)$$

So by knowing the value of the derivative at that point, the tangent to the solution curve can be easily obtained as

$$y = y_0 + f(x_0, y_0)(x - x_0). \qquad (8)$$

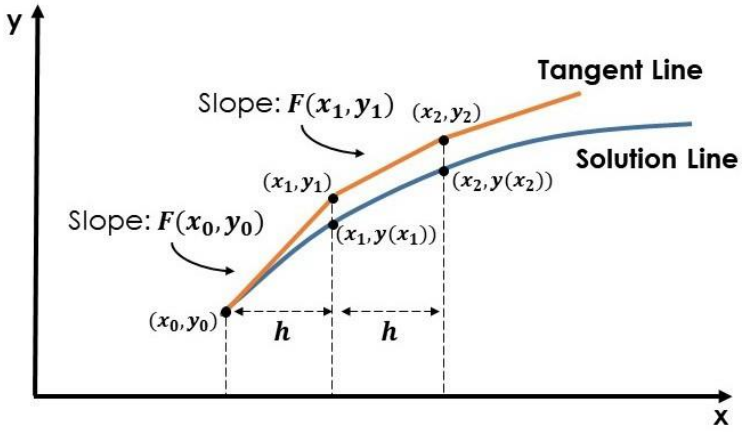



Figure 3: Euler's method approximation.

If $x_1$, a close enough point to $x_0$, is chosen, then the point $y_1$ on the tangent line is directly close to the actual value of the solution at $x_1$. We can calculate $y_1$ by

$$y_1 = y_0 + f(x_0, y_0)(x_1 - x0). \tag{9}$$

By similar manner, we can construct our approximation. So, let's hope that $y_1$ is a good approximation to the solution and draw a new line at point $(x_1, y_1)$ whose slope is $f(x_1, y_1)$.

$$y_2 = y_1 + f(x_1, y_1)(x_2 - x_1) \tag{10}$$

In a similar fashion,

$$y_2 = y_1 + f(x_1, y_1)(x_2 - x_1) \tag{11}$$

$$y_3 = y_2 + f(x_2, y_2)(x_3 - x_2) \tag{12}$$

$$y_4 = y_3 + f(x_3, y_3)(x_4 - x_3) \tag{13}$$

In general, if we have $x_{n-1}$ and the approximation to the solution at that point, $y_{n-1}$, we can find the approximation $y_n$ at point $x_n$ as

$$y_n = y_{n-1} + f(x_{n-1}, y_{n-1})(x_n - x_{n-1}). \tag{14}$$

If we define $f_{n-1} = f(x_{n-1}, y_{n-1})$ and the step size $h = x_n - x_{n-1}$, the approximation becomes

$$y_n = y_{n-1} + hf_{n-1} \tag{15}$$

Using a uniform step size $h$, the pseudo-code for Euler's method can be summarised as

1. **define** $f(x, y)$
2. **input** $x_0$ and $y_0$
3. **input** step size $h$ and the number of steps $n$
4. **for** j from 1 to $n$ do
   (a) $m = f(x_{j-1}, y_{j-1})$
   *(b)* $x_j = x_{j-1} + h$
   *(c)* $y_j = y_{j-1} + hm$
   *(d)* **save** $y_j$ and $x_j$
5. **end**

### 2.1.1 MATLAB Tutorial

Consider the following differential equation

Consider the following differential equation

$$\frac{df}{dt} + 2f = 2 - \exp^{-4t}, \qquad f_0 = 1 \tag{16}$$

Let's try to solve the given differential equation using Euler's numerical method. Then compare approximation with the actual solution. First, find the exact solution by solving the differential equation

```
>> syms f(t) % define the symbolic function
>> ode = diff(f,t) == -2*f+2-exp(-4*t); % define the differential equation
>> conds = f(0)==1; % set the initial condition
>> fSol = dsolve(ode, conds); % solve the differential equation
>> td = [0:0.01:5]; % set the display interval
>> % plot the obtained result in the chosen interval
>> figure(1)
>> plot(td , subs(fSol, t,td ),'Linewidth',2,'DisplayName','Exact Solution')
>> hold on
>> grid on
>> ylim([0.6 1])
```

Then, write the code to approximate the solution using Euler's method.

```
>> h = 0.01; % set step size
```

```
>> n = 200; % set number of steps

>> x = zeros(1,n); % allocate memory for x
>> y = zeros(1,n); % allocate memory for y
>> x(1)= 0; % initial condition, x 0
>> y(1)= 1; % initial condition, y 0
>> for i=1:n
>> m = -2*y(i)+2-exp(-4*x(i)); % evaluate f(x,y)
>> y(i+1) =y(i)+h*m; % find and save next y
>> x(i+1)=x(i)+h; % find and save next
>> end
>> % plot the result
>>       plot(x,y,'--','Linewidth',2,'DisplayName',['h=',sprintf('%.2f',h)])
>> xlim([0, 5])
>> legend
>> xlabel('t')
>> ylabel('f')
```

Exact solution and the approximations for different steps sizes are given in Figure 4.

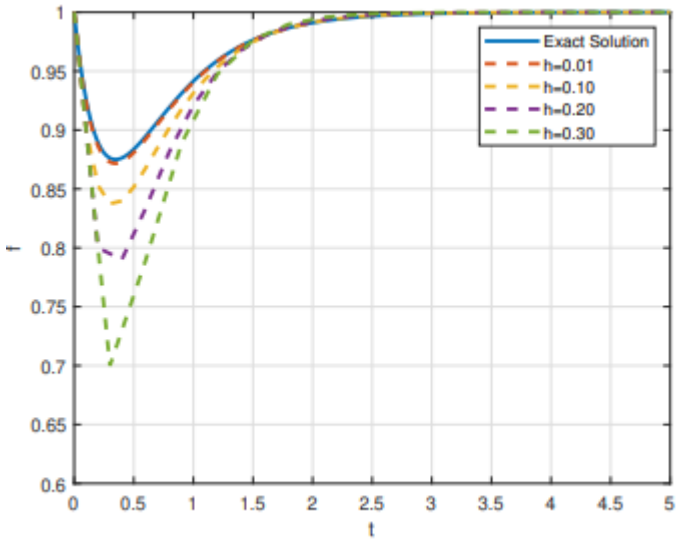



Figure 4: Exact solution compared with Euler's method using different step sizes (*h*).

### 2.1.2 Question 2

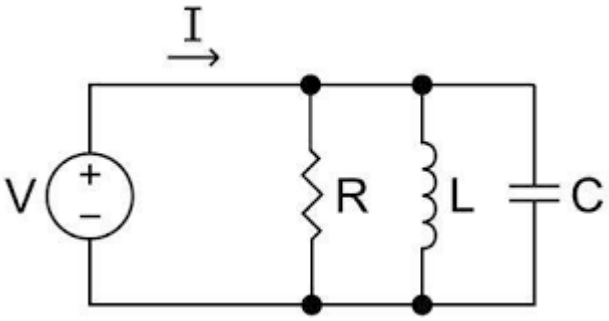
Use the Euler's Method to find the approximation to the solution of the following differential equation,

$$y' = 5\, e^{\frac{t}{2}} \cos(5t\,) - \frac{1}{2} e^{\frac{t}{2}} \sin(5t\,) + y$$

Given the initial condition $y(0) = 0.$ at time t from 0 to 10 with increment of 1, e.g. (0, 1, … 10). Use 5 different values of h for this approximation like (h=0. 01, ...). In the end, compare and plot the results of the approximations solution with the exact solution. (For example, first, choose h=0.001 and find the exact and approximate solution for t from 0 to 10 and plot it, Same goes for the other five values of h, you are free to choose any value of h). Furthermore, at the end, report which value of h gives you the best results.

### 2.1.3 Question 3

Differential equations can be used to describe the behavior of electrical circuits consisting of resistors, capacitors, and inductors. Consider the following RLC circuit,




With resistance R (ohms), inductance L (henry) and the capacitance C (Farads). Voltage at time t is defined by V(t). The current through circuit using the nodal analysis and then by differentiating with respect to t in the form of differential equation is

$$\frac{di}{dt} = C\,\frac{d^2V(t)}{dt^2} + \frac{1}{R}\frac{dV(t)}{dt} + \frac{1}{L}V(t)$$

Where R, C and L are constant 2 ohms, 0.1 farads and 1.5 henry, respectively. Where the voltages in the circuit are

$$V(t) = e^{-\frac{1}{2L}RC\pi t}$$

Given the initial condition $i(0) = (0)$, find the current following in the circuit when the times sweeps from 0 to 100 with the step size of 0.2, e.g. (0, 0.2, … 100). Also plot the solution against the time axis.

## 2.2 Runge-Kutta Method

Runge Kutta method is a popular iteration method of approximating solution of ordinary differential equations. The Runge Kutta method is a based-on solution procedure of initial value problems in which the initial conditions are known. Most widely known Runge Kutta family is the fourth order one which is generally referred as RK4.

Consider an initial value problem

$$\frac{dy}{dx} = f(x, y), \qquad y(x_0) = y_0 \tag{18}$$

$$\left.\frac{dy}{dx}\right|_{x=x_0} = f(x_0, y_0) \tag{19}$$

$$y = y_0 + f(x_0, y_0)(x - x_0) \tag{20}$$

Here $y$ is an unknown function of $x$, which we would like to approximate. $f(x, y)$ is the rate at which $y$ changes and it is a function of $y$ and $x$. $(x - x_0)$ is a step size which is represented by $h$. In stead of $f(x_0, y_0)$, there will be a new average slope to estimate $y$. The new slope is represented by $T_4(x_0, y_0, h)$ and depends on the initial conditions of $x$, $y$, and the step size $h$.

We can approximate the solution using Runge Kutta method by picking the step size $h > 0$ and defining

$$y_{n+1} = y_n + hT_4(x_n, y_n, h) \tag{21}$$

$T_4$ combines the average of four different slopes $k_1$, $k_2$, $k_3$, and $k_4$.

$k_1$ is the increment based on the slope at the beginning of the interval, using $y$;

$k_2$ is the increment based on the slope at the midpoint of the interval, using $y$ and $k_1$;

$k_3$ is again the increment based on the slope at the midpoint, but now using $y$ and $k_2$;

$k_4$ is the increment based on the slope at the end of the interval, using $y$ and $k_3$.

So, the slope became

$$T_4 = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{22}$$

$$k_1 = f(x, y) \tag{23}$$

$$k_2 = f\left(x + \frac{h}{2}, y + \frac{h}{2}k_1\right) \tag{24}$$

$$k_3 = f\left(x + \frac{h}{2}, y + \frac{h}{2}k_2\right) \tag{25}$$

$$k_4 = f(x + h, y + hk_3) \tag{26}$$

Using these information, we can develop the pseudo-code of Runge Kutta method as

1. **define** step size $h$, range of $x$, and the initial conditions(s)
2. **define** $f(x, y)$
3. **write** a loop to find $k_1$, $k_2$, $k_3$, and $k_4$
4. **end**

### 2.2.1 MATLAB Tutorial

Lets try to solve the following differential equation using Runge Kutta method.

$$\frac{dy}{dx} = y - x^2 + 1, \qquad 0 < x < 4, \qquad y(0) = 0.5 \tag{27}$$

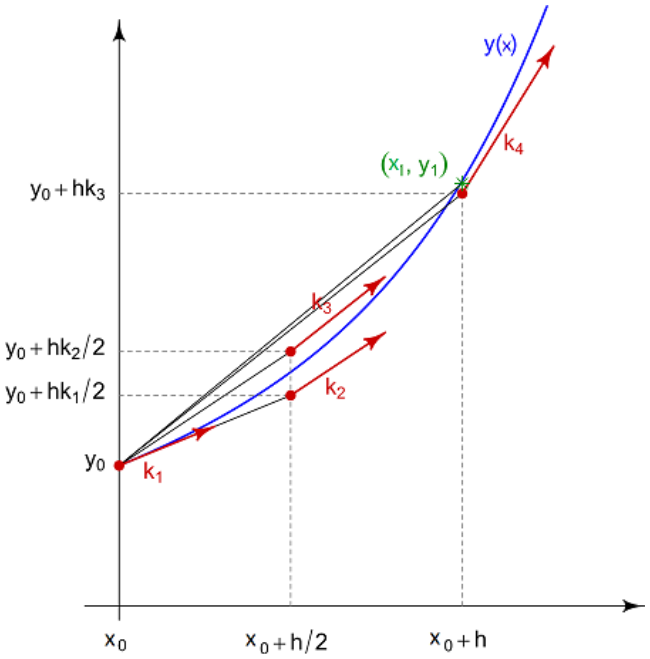



Figure 5: Extraction of Slopes $k_1$, $k_2$, $k_3$, and $k_4$.

Fisrt, lets find the exact solution by using MATLAB ode45 function.

```
>> xspan = [0 4];
>> y0 = 0.5;
>> [x,y] = ode45(@(x,y) y-x^2+1, xspan, y0);
```

```
>>  plot(x,y,'-o','Linewidth',2,'DisplayName','Exact  Solution')
>> hold  on
```

Then, lets write the code approximating the solution according to Runge Kutta method using the given pseudo-code

```
>> a = 0; % x interval we are solving over
>> b  =  4;
>> N  =  1000;  %number  of  steps
>> x = zeros(1, N); %reallocate memory for these variables - this is preferred but is
not required to
>> y  =  zeros(1,  N);
>> y(1) = 0.5; %initial value of y.
>> h = (b - a)/N; %step size
>> x(1) = a; %initial value of x.
>> F = @(t, y) y - t^2 + 1; %thing  we  are solving.
>> for i = 1:(N-1)
>> %this part is the Runge Kutta Method.
>> K1  =  F(x(i), y(i));
>> K2 = F(x(i) + 0.5*h, y(i) + 0.5*h*K1);
>> K3 = F(x(i) + 0.5*h, y(i) + 0.5*h*K2);
>> K4 = F(x(i) + h, y(i) + h*K3);
>> T4 = (K1 + 2*K2 + 2*K3 + K4)/6;
>> y(i+1) = y(i) + h*T4;
>> x(i+1) = a + i*h;
>> end
>> plot(x, y,'Linewidth',2,'DisplayName','Runge Kutta Method')
>> hold off
>>  legend('location','northwest')
>> grid on
```

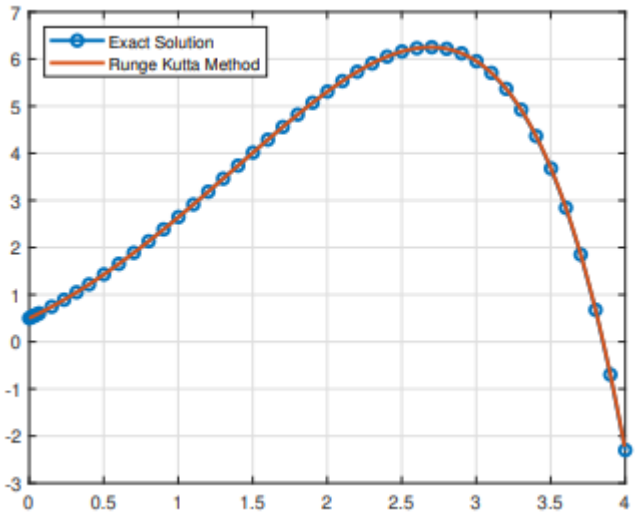
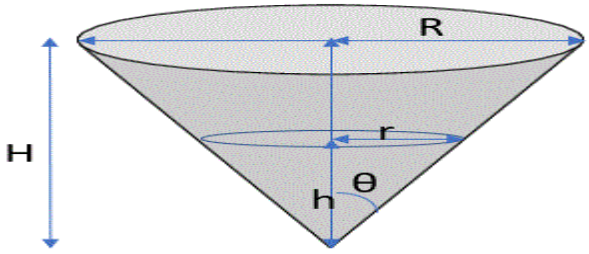



Figure 6: Exact solution compared with Runge-kutta Method.

### 2.2.2 Question 4

Consider an inverted conical tank which is filled with water, the water drains from the hole of radius r called orifice with the rate given by the following rate,




$$\frac{dy}{dt} = -\frac{1}{2}\pi r^2\sqrt{\frac{3}{2}g}\,\frac{y^{\frac{3}{2}}}{A(y)}$$

A(y) is the area of cross section of the tank y unit above the hole/orifice, the height of the liquid in the cone is denoted by the y from the vertex of the cone. Use Use the Runge-Kutta method of order four to find the following

a) The water level in the conical tank after 10 min with h = 20 s

b) Time required by the tank to be empty, within 1 min.

Supposing the radius r=0.5 ft, having the gravitational force g= 42 ft/$s^2$ while the tank has an initial water level of 8 ft and initial volume of 512(π/3) $ft^3$

### 2.2.3 Question 5

Use the Runge Kutta method of order 4 (RK4) to find the solution of the following differential equation

$$\frac{dy}{dt} = (t + y)\sin(t + y)$$

With the initial condition $y(0) = 5$

a) Consider the h=0.3 and plot the solution of the above equation as time t changes from 0 to 10.

b) In part a, you will observe the curve obtained is not smooth, do the necessary steps to make the curve smooth, and plot it again. Also, comment which measures helped you in achieving a smooth curve and why.