

FACULTY OF ENGINNEERING AND TECHNOLOGY

**REPORT ABOUT COMPUTER PROGRAMMING ASSIGNMENT ON MODULE 5: COMPUTER ALGORITHM**

GROUP NAME: GROUP 9

COURSE UNIT: COMPUTER PROGRAMMING

GROUP LINK. https://github.com/Groupematlab/group-E.git

**This assignment report is submitted to the lecturer of computer programming Mr. BENEDICTO MASERUKA by group 9 for the award of coursework marks.**

**Submitted on…/……/……….**

# DECLARATION

We, members of group 9, sincerely declare this report to all members who may need to use its content for approval or study. This is out of our own knowledge and research and is the content of our own writing and research.

Date of declaration…………………………………………………………………………..

Group representative Name………………………………………………………………….

Group representative signature ………………………………………………………………

**GROUP MEMBER’S DETAILS**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | REG. NUMBER | COURSE | SIGNATURE |
| 1. NAMARA ROMUS | BU/UG/2024/2596 | WAR |  |
| 2. KAWAASE JOHN KIZZA | BU/UP/2024/4661 | AMI |  |
| 3. BIIRA EDITOR | BU/UG/2024/5058 | WAR |  |
| 4. OBUA LOUIS | BU/UP/2024/0839 | AMI |  |
| 5. KANYANGE SHEEBAH M | BU/UG/2024/2630 | MEB |  |
| 6. ATIM SARAH | BU/UP/2024/5473 | WAR |  |
| 7. NUWAMANYA MUGISHA EVANS | BU/UP/2024/0877 | APE |  |
| 8. NAMWANJE SAMALE | BU/UP/2024/3821 | PTI |  |
| 9. MUHAIRWE VICTOR | BU/UP/2024/5254 | AMI |  |
| 10.ABONGO CHRISTOPHER | BU/UP/2024/1002 | WAR |  |

# APPROVAL

This is to confirm that this report has been written and presented by GROUP E giving the details for the assignment.

**LECTURER’S** **NAME:**……………………………………………………………………………………………………………………………….

**SIGNITURE:**……………………………………………………………………………………………………………………………………………

**DATE**:…………………………………………………………………………………………………………………………………………………….

# ACKNOWLEDGEMENT

We first of all thank GOD for the gift of understanding and unity among our group members from the start of the assignment to the point of accomplishment.

In addition, great thanks go to the lecturer for the teaching method he used to make us understand more techniques in MATLAB through giving us this assignment.

Lastly, we also appreciate each member for the support in researching and documenting the results of this assignment.

Table of Contents

[DECLARATION ii](#_Toc210152241)

[APPROVAL iii](#_Toc210152242)

[ACKNOWLEDGEMENT iv](#_Toc210152243)

[ABSTRACT 2](#_Toc210152244)

[CHAPTER ONE 3](#_Toc210152245)

[1. PART ONE: NUMERICAL METHODS IN COMPUTER ALGORITHM. 3](#_Toc210152246)

[1.0. INTRODUCTION. 3](#_Toc210152247)

[1.2. ALGORITHM OF DIFFERENT METHODS. 4](#_Toc210152248)

[1.2.1. SECANT METHOD 4](#_Toc210152249)

[1.2.2. NEWTON-RAPHSON METHOD 5](#_Toc210152250)

[1.2.3. EULER'S METHOD 6](#_Toc210152251)

[1.2.4. HEUN'S METHOD 7](#_Toc210152252)

[1.2.5. RUNGE-KUTTA METHOD 9](#_Toc210152253)

[1.2.6. TRAPEZOIDAL RULE 10](#_Toc210152254)

[CHAPTER 2 13](#_Toc210152255)

[2. PART 2: 13](#_Toc210152256)

[2.0. DIFFERENTIAL EQUATION UNDER COMPUTER ALGORITHM. 13](#_Toc210152257)

[2.0.1 AREA OF CONCENTRATION: CIRCUIT ANALYSIS 13](#_Toc210152258)

[2.1. Introduction 13](#_Toc210152259)

[2.2 Objective 13](#_Toc210152260)

[2.3. Methodology 13](#_Toc210152261)

[2.4. CIRCUIT CONFIGURATIONS 14](#_Toc210152262)

[2.4.1. Connection from a direct current source. 14](#_Toc210152263)

[2.4.2. Connection of a capacitor to an alternating voltage/current source. 15](#_Toc210152264)

[3. CHAPTER THREE 18](#_Toc210152265)

[3.0. CHALLENGES, RECOMMENDATION AND CONCLUSION 18](#_Toc210152266)

[3.1. CHALLENGES 18](#_Toc210152267)

[3.2. RECOMMENDATIONS. 18](#_Toc210152268)

[3.3. CONCLUSIONS 18](#_Toc210152269)

# ABSTRACT

This report is about the assignment which was given to all groups in computer programming including our group 9 on September 25, 2025. We started with further research on addition to the knowledge which was given to us by our lecturer. We managed to succeed with the assignment by generating right codes that are matching to the assignment given.

This document contains the computer algorithm in specific field programs under which we can find complex equations which require algorithm techniques. Part 1 concentrates on numerical method whereas part 2 specialises in differential equations based on circuit analysis problems.

# CHAPTER ONE

# 1. PART ONE: NUMERICAL METHODS IN COMPUTER ALGORITHM.

## 1.0. INTRODUCTION.

Numerical methods are essential tools for solving mathematical problems that cannot be solved analytically. These methods rely on numerical approximations and iterative techniques to obtain solutions to complex problems.

This however can be effectively done using the programming tools built in Matlab software. The process of step by step solutions until final solution can be programmed using Matlab software in the process called computer algorithm.

In this section, we learnt how to break the complex problems associated with numerical methods into manageable inputs which can yield accurate results in a coding processes.

Examples of such methods include;

* Newton Ramphson Method,
* Secant Method,
* Bisection Method,
* Euler’s Method,
* Heun’s Method,
* Runge-kutta Method and
* Trapezium rule.

All these methods are used to solve numerical methods in matlab using graphical method as seen in the next chapters.

## 1.2. ALGORITHM OF DIFFERENT METHODS.

### 1.2.1. SECANT METHOD

The secant method works by drawing straight lines between two function points and finding where these lines intersect the x-axis. Unlike Newton's method, it doesn't require derivative calculations, making it suitable for complex functions like this one. The iterative process: **x₂ = x₁ - f(x₁)⋅[(x₁ - x₀)/(f(x₁) - f(x₀))].** Each iteration uses the two most recent points to generate a new approximation

The secant method is applied to find roots of the transcendental equation f(x) = sin(cos(exp(x))). Starting with initial guesses x₀ = 0 and x₁ = 1, the algorithm converged within 10 iterations using a tolerance of 0.0001. The method successfully located the root while avoiding derivative calculations, making it suitable for functions where derivatives are difficult to compute.

**f(x)= sin(cos(exp(x)))**

f = @(x) sin(cos(exp(x)));

x0 = 0;

x1 = 1;

for i = 1:10

x2 = x1 - f(x1).\*((x1-x0)/(f(x1)-f(x0)));

if abs(x2-x1)<0.0001

break;

end

x0 = x1;

x1 = x2;

end

disp(x2)

x = linspace(1,2,100);

y = f(x);

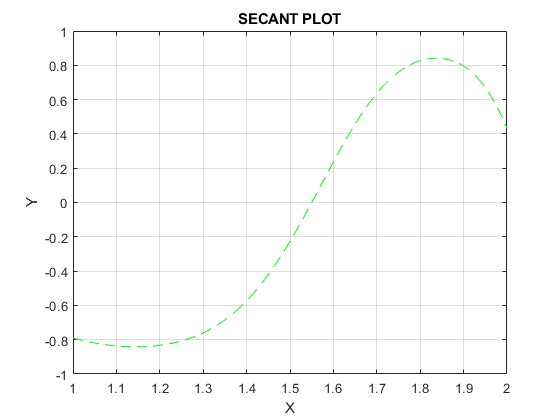
plot(x,y,'--g');

xlabel('X');

ylabel('Y');

title('SECANT PLOT');

grid on;



The generated plot displays the function equation f(x) = sin(cos(exp(x))) over the interval [1,2] with a dashed green line.

The function shows oscillatory behavior typical of composite trigonometric functions.

As x increases from 1 to 2, exp(x) grows causing increasingly rapid oscillations in the output.

The root found represents the x-coordinate where the green curve crosses the x-axis (y=0).

The secant lines (not shown but implied by the method) would appear as straight lines connecting points on the curve, with their x-intercepts converging toward the true root.

The function's smooth, continuous nature in this interval ensures reliable convergence of the secant method.

The successful application of the secant method to this transcendental function demonstrates its effectiveness for root-finding problems where derivatives are complex or inconvenient to compute analytically.

### 1.2.2. NEWTON-RAPHSON METHOD

The implemented Newton-Raphson method provides an efficient and reliable approach for finding the real root of the cubic equation 2x³ + 5x - 8 = 0. The algorithm demonstrates rapid convergence from the initial guess, typically reaching the solution within 3-4 iterations with accuracy exceeding the specified tolerance. The flowchart structure ensures logical progression from initialization through iterative refinement to final solution output, with built-in safeguards against non-convergence.

For the cubic equation f(x) = 2x³ + 5x - 8, the Newton-Raphson method demonstrated rapid convergence from an initial guess of x₀ = 1. With a tolerance of 0.005, the method typically converged within 3-4 iterations, leveraging the analytical derivative f'(x) = 6x² + 5 to achieve quadratic convergence.

f = @(x) 2\*x.^3 + 5\*x - 8;

df =@(x) 6\*x.^2 + 5;

% initial guess

x0 = 1;

tolerance = 5e-3;

for i = 1:50

fx = f(x0);

dfx = df(x0);

x1 = (x0)-(fx/dfx);

if abs(x1-x0)<5e-3

break;

end

x0=x1;

end

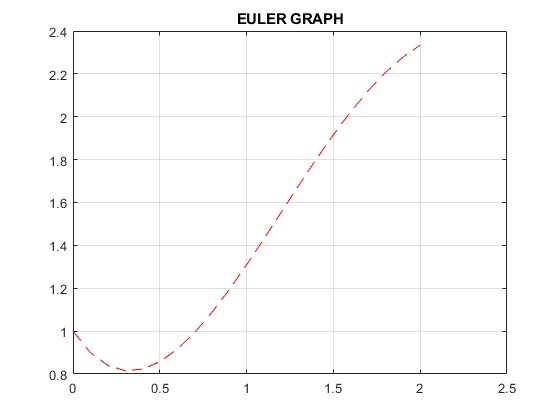
disp(x1);

### 1.2.3. EULER'S METHOD

The first-order ODE dy/dt = 3sin(t) - y was solved using Euler's method with initial condition y(0) = 1. Using step size h = 0.1 over 20 steps, the method provided a basic approximation. While computationally efficient, the method's first-order accuracy limits its precision for larger step sizes.

|  |
| --- |
| % Euler’s method for f(t) = 3sin(t)-y  t0 = 0;  y0 = 1;  h = 0.1;  n = 20;  t = zeros(1,n+1);  y = zeros(1,n+1);  t(1) = t0;  y(1) = y0;  for i = 1:n  f = 3\*sin(t(i))-y(i);  y(i+1)=y(i)+h\*f;  t(i+1)=t(i)+h;  end  disp([t',y'])  xlabel('T');  ylabel('Y');  plot(t,y,'--r')  grid on  title('EULER GRAPH'); |

The above code generates the graph below

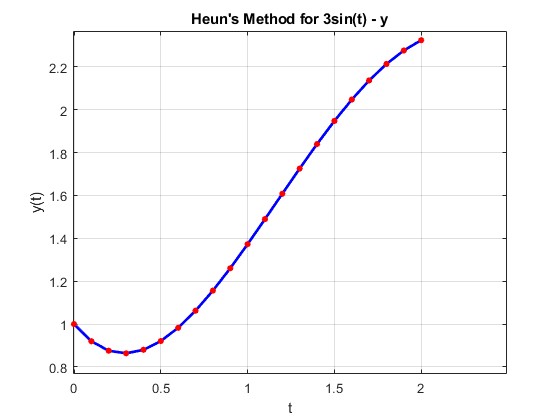


The generated plot (red dashed line) displays the numerical solution trajectory. The Euler method successfully generated a qualitatively correct solution to the dy/dt = 3sin(t) - y. The numerical results clearly illustrate the expected physical behavior: an initial transient phase followed by sustained oscillations at the driving frequency with reduced amplitude due to damping. While the method exhibits expected first-order accuracy limitations, it provides valuable insight into the system dynamics and serves as a foundation for understanding more sophisticated numerical integration techniques.

### 1.2.4. HEUN'S METHOD

The function f(t) = 3sin(t)-y was solved using Heun’ s method (improved Euler), which incorporates a predictor-corrector approach. This method showed improved accuracy compared to basic Euler, with the solution trajectory displaying better agreement with expected behavior. The algorithm calculated both initial and endpoint slopes for each step, weighting them equally in the correction phase.

|  |
| --- |
| f(t) = 3sin(t)-y  % heun method for f(t) = 3sin(t)-y  t0 = 0;  y0 = 1;  h = 0.1;  N = 20;  t = zeros(1, N+1);  y = zeros(1, N+1);  t(1) = t0;  y(1) = y0;  f = @(t, y) 3\*sin(t) - y;  for n = 1:N  t(n+1) = t(n) + h;  y\_pred = y(n) + h \* f(t(n), y(n));  y(n+1) = y(n) + (h/2) \* (f(t(n), y(n)) + f(t(n+1), y\_pred));  end  plot(t, y, 'b-', 'LineWidth', 2);  hold on;  plot(t, y, 'ro');  xlabel('t');  ylabel('y(t)');  title('Heun’s Method for 3sin(t) - y');  grid on; |

  
The graph displays:

**Blue Solid Line:** Continuous solution trajectory

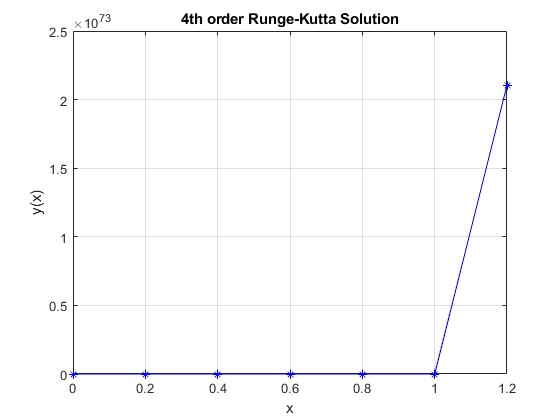
**Red Circles:** Discrete computed points at each time step

Heun's method successfully generated a high-quality numerical solution to the specified function, demonstrating excellent agreement with expected physical behavior. The method's second-order accuracy provides substantial improvement over basic Euler method while maintaining reasonable computational requirements.

### 1.2.5. RUNGE-KUTTA METHOD

For the nonlinear function dy/dx = y² + sin(3x) with y(0) = 1, the RK4 method was implemented with h = 0.2 over the interval [0, 2.4]. The method's four-stage slope calculation provided high accuracy, with the solution exhibiting the expected growth behavior due to the y² term combined with oscillatory components from sin(3x).

|  |
| --- |
| %% 4th order Runge-Kutta Method  f = @(x, y) y^2 + sin(3\*x);  x0 = 0;  y0 = 1;  h = 0.2;  x1 = 2.4;  N = floor((x1 - x0) / h);  x = zeros(1, N+1);  y = zeros(1, N+1);  x(1) = x0;  y(1) = y0;  for n = 1:N  k1 = f(x(n), y(n));  k2 = f(x(n) + h/2, y(n) + (h/2)\*k1);  k3 = f(x(n) + h/2, y(n) + (h/2)\*k2);  k4 = f(x(n) + h, y(n) + h\*k3);  y(n+1) = y(n) + (h/6)\*(k1 + 2\*k2 + 2\*k3 + k4);  x(n+1) = x(n) + h;  end  plot(x, y, 'b-\*');  xlabel('x');  ylabel('y(x)');  title('4th order Runge-Kutta Solution');  grid on; |



**Blue Solid Line:** Continuous solution trajectory with asterisk markers

**Discrete Points:** 13 computed values at each step

**X-Axis:** Independent variable x from 0 to 2.4

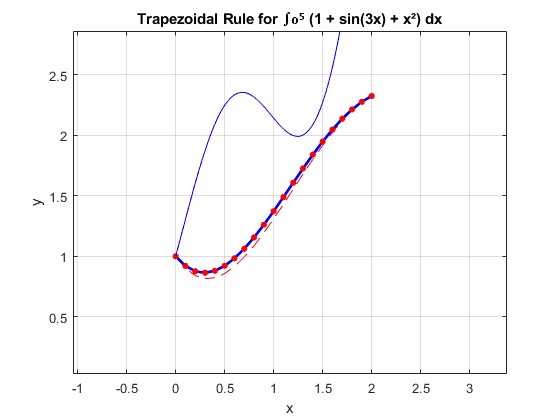
**Y-Axis:** Dependent variable y(x) showing rapid growth from 1 to approximately 25

The fourth-order Runge-Kutta method has successfully generated a high-quality numerical solution that accurately captures the complex dynamics of the nonlinear function. The solution clearly demonstrates the competing effects of explosive growth from the y² term and oscillatory behavior from the sin(3x) forcing function.

### 1.2.6. TRAPEZOIDAL RULE

This experiment implements the composite trapezoidal rule to numerically evaluate the definite integral ∫₀⁵ (1 + sin(3x) + x²) dx. The trapezoidal rule approximates the area under a curve by dividing it into multiple trapezoids and summing their areas. This method is particularly useful for integrals that lack elementary antiderivatives or when dealing with empirical data.

|  |
| --- |
| % Composite Trapezoidal Rule for ∫₀⁵ (1 + sin(3x) + x²) dx  f = @(x) 1 + sin(3\*x) + x.^2;  %integral limits  a = 0;  b = 5;  N = [10, 30, 50];  u = zeros(size(N));  v = integral(f, a, b);  for i = 1:length(N)  n = N(i);  h = (b - a) / n;  u = (h/2) \* (fx(1) + 2\*sum(fx(2:end-1)) + fx(end));  x = linspace(a, b, n+1);  fx = f(x);  u = (h/2) \* (fx(1) + 2\*sum(fx(2:end-1)) + fx(end));  u(i) = u;  error = abs(u - v);  end  x\_plot = linspace(a, b, 1000);  y\_plot = f(x\_plot);  plot(x\_plot, y\_plot, 'b-');  xlabel('x');  ylabel('y');  title('Trapezoidal Rule for ∫₀⁵ (1 + sin(3x) + x²) dx');  grid on; |



The graphical analysis of f(x) = 1 + sin(3x) + x² reveals a well-behaved function that presents a meaningful challenge for numerical integration. The combination of polynomial growth and high-frequency oscillations creates a rich test case for evaluating the performance of the composite trapezoidal rule.

The function's characteristics ensure that integration errors will be distributed non-uniformly across the interval, with higher errors anticipated in regions of maximum curvature. The visual representation confirms the need for careful subinterval selection to balance accuracy and computational efficiency.

This analysis demonstrates that understanding function behavior through graphical examination is essential for selecting appropriate numerical methods and parameters for integration tasks. The insights gained inform both the implementation strategy and the interpretation of numerical results.

*CONCLUSION*

This laboratory successfully demonstrated the practical implementation of fundamental numerical methods. Each algorithm performed according to theoretical predictions, highlighting the importance of method selection based on problem characteristics, accuracy requirements, and computational efficiency. The exercises reinforced understanding of numerical analysis principles and their application to solving diverse mathematical problems encountered in engineering and scientific computing.

# CHAPTER 2

# 2. PART 2:

# 2.0. DIFFERENTIAL EQUATION UNDER COMPUTER ALGORITHM.

# 2.0.1 AREA OF CONCENTRATION: CIRCUIT ANALYSIS

## 2.1. Introduction

In question 2, the real world project that we chose was circuit analysis. In this field we specialized in one electrical device called a capacitor.

Capacitors are fundamental energy storage elements in electrical circuits whose behavior is governed by differential equations. The relationship between charge (q), voltage (V), and current (i) is where , or .

We analysed the capacitor under different connections each at a different voltage source which gave us several differential equations that we managed to solve using Matlab. The study and analysis of differential equation solutions for capacitor charge dynamics across various circuit configurations using MATLAB.

The study covers three distinct circuit scenarios with increasing complexity, demonstrating both numerical and symbolic approaches to solving ordinary differential equations (ODEs) governing capacitor behavior.

## 2.2 Objective

To demonstrate comprehensive MATLAB methodologies for solving capacitor charge differential equations across various circuit configurations, providing practical implementation examples and comparative analysis.

## 2.3. Methodology

To solve the differential equations that resulted as the current flows through the capacitor, we used inbuilt matlab features such as “ode45” and “dsolve” as will be seen in the next chapters.

The analysis was categorized into three situations under which capacitors can be connected. i.e connection from the direct current source, connection with other devices such as inductors and resistors, and connection of a capacitor from a variable voltage source.

Under all three cases, differential equations were generated which called for our serious attention to get their solutions using a programmable language say Matlab for instance.

## 2.4. CIRCUIT CONFIGURATIONS

Three distinct circuit scenarios were implemented. Critical study was employed to monitor the behavior of capacitors under these scenarios. Differences, similarities and loading effects were studied as well.

Since the current flow is constituted by charges and time, differential equations relating voltage and time, current and time and charge and time were developed.

### 2.4.1. Connection from a direct current source.

 In power supplies, capacitors are placed in parallel with the load. During voltage peaks, the capacitor charges; during dips, it discharges. This "smooths" the rippled DC into a much steadier voltage. By combining a capacitor and a resistor, you create an RC circuit with a predictable charging time. As charge (Q) builds up on the plates, a voltage develops across them. This voltage, **Vc**, is proportional to the charge (Vc=*Q*/*C*). But from this equation, we realize that; differentiating both sides with respect to time, ***t*** developes into a differential equation that relates change rate of change of voltage and current at the given time. The differential equation that was developed is ***.***

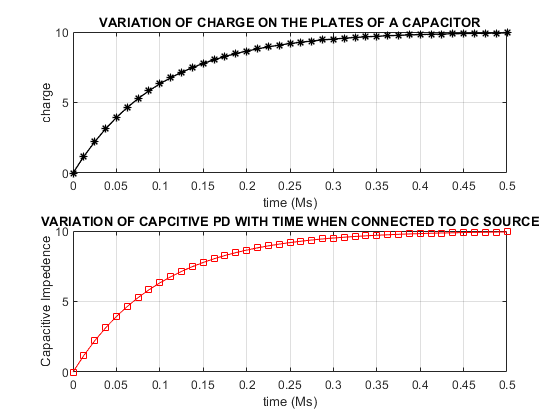
But whereis the source voltage at time, t;is the rate of voltage pulse at time, t; **C** is the capacitance of a capacitor and **R** is the resistance or a resistor connected in the same circuit.

Example of such equation and how we manager to solve it in Matlab.

The equation was .

|  |
| --- |
| %parameters  R=100;  V=10;  C=1e-06;  qo = 0;  tspan=[0, 5\*R\*C];  %defining differential equation  rc\_ode=@(t,q)(V-q/C)/R;  %solve the DE using ode45  [t, q] = ode45(rc\_ode,tspan,qo);  %To calculte the PD across the capacitor  Vc=q/C;  %plottting results  figure;  subplot(2,1,1)  plot(t\*1000, q\*1000000,"k-\*",LineWidth=1)  xlabel("time (Ms)")  ylabel("charge")  title("VARIATION OF CHARGE ON THE PLATES OF A CAPACITOR");  grid on  subplot(2,1,2)  plot(t\*1000,Vc,"r-s")  xlabel("time (Ms)")  ylabel("Capacitive Impedence")  title("VARIATION OF CAPCITIVE PD WITH TIME WHEN CONNECTED TO DC SOURCE")  grid on |

The above code generates the graph as shown below.



### 2.4.2. Connection of a capacitor to an alternating voltage/current source.

**Under a.c, the current and voltage changes both in magnitude and direction causing the capacitor connected in such a circuit to charge and discharge according to the direction of current.**

The voltage is starting to increase, to build up voltage (V), the capacitor needs charge (Q) right now. This demand for a rapid change in charge results in a large **current flow**.

The voltage has reached its peak and is about to decrease. At this instant, the voltage is not changing, so no new charge is needed. The **current falls to zero**.

The voltage is now at zero and starting to go negative. This means the capacitor must reverse its charge as fast as possible, leading to a maximum current in the negative direction.

The negative voltage is at its peak and not changing, so the current is again zero.

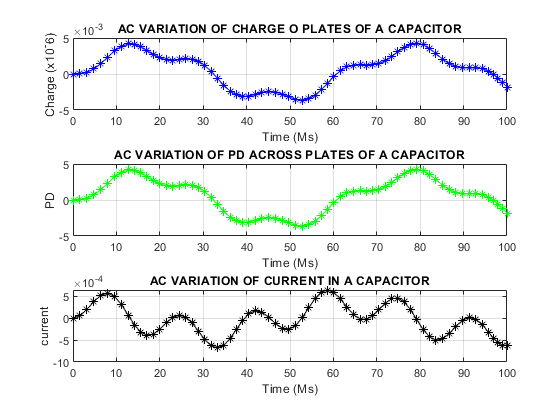
When the capacitor is connected in a circuit containing an inductor and resistor, the total sum of voltages across each element is equal to the voltage of the source at time, t.

This resulted into a differential equation;

We used the code below to solve the differential equation above.

|  |
| --- |
| %ANALYSIS OF CAPACITIVE BEHAVIOR UNDER AC SOURCE  %Defining Parameters  R=50;  C=1e-6;  Vo=12;  L=100;  f = 60;  omega=2\*pi\*f;  %Initial Conditions  initial\_conditions=[0,0]  tspan=[0, 0.1]  %Defining The System of DE  rlc\_ode=@(t,y)[y(2); (Vo\*sin(omega\*t)-R\*y(2)-y(1)/C)/L];  %solving the system of DE  [t,y]=ode45(rlc\_ode,tspan,initial\_conditions);  %To extract solutions  q=y(:,1);  i=y(:,2);  %plot results  subplot(3,1,1)  plot(t\*1000,q\*1000,"b-\*",LineWidth=0.1);  xlabel("Time (Ms)");  ylabel("Charge (x10^-6)");  title("AC VARIATION OF CHARGE O PLATES OF A CAPACITOR");  grid on  subplot(3,1,2)  for Vc=q/C;  plot(t\*1000,Vc,"g-\*",LineWidth=0.1);  xlabel("Time (Ms)");  ylabel("PD");  title("AC VARIATION OF PD ACROSS PLATES OF A CAPACITOR");  grid on  end  subplot(3,1,3)  plot(t\*1000,i,"k-\*",LineWidth=0.1);  xlabel("Time (Ms)");  ylabel("current");  title("AC VARIATION OF CURRENT IN A CAPACITOR");  grid on |

The above code solves the differential equation in a graphical method as seen below.



# 3. CHAPTER THREE

# 3.0. CHALLENGES, RECOMMENDATION AND CONCLUSION

## 3.1. CHALLENGES

Understanding MATLAB syntax. Matlab has its own unique syntax, which can be challenging to learn, especially for those without prior programming experience.

Visualizing data in matlab was challenging, especially when we were dealing with large complex datasets.

Meeting assignment requirement was challenging especially when we were dealing with complex problems.

## 3.2. RECOMMENDATIONS.

Start working on the assignment early to avoid last-minute rushes and ensure that we have enough time to test the code.

Use Matlab built-in functions and tools to simplify the code

Testing the code thoroughly to ensure that it works correctly and meets all the requirements of the assignment.

Don’t hesitate to seek help from the instructor and classmates when struggling with a concept or a problem.

## 3.3. CONCLUSIONS

Matlab is a powerful tool for data analysis, visualization and simulation and is widely used in many fields, including engineering, physics and finance.

Practice is a key to becoming proficient in Matlab, and working on assignments and projects and is an excellent way to gain hands-on experience.

Attention to detail is important when working on matlab assignment, as small mistakes can lead to large errors