FACULTY OF ENGINEERING & TECHNOLOGY

A REPORT ABOUT USING THE ALGORITHM DEVELOPEMENT, CONTROL STRUCTURE AND MODULES 1-4 ON PRACTICAL REAL-WORLD PROBLEM

PRESENTED TO THE COMPUTER PROGRAMMING COURSE LECTURER

MR. MASERUKA BENDICTO

BY GROUP 17

GROUP MEMBERS

	NAME	REG NUMBER	PROGRAM
1	AHAISIBWE CHRISTOPHER	BU/UP/2024/0824	AMI
2	GIFT EMMANUEL	BU/X/2024/3250	WAR
3	ADWONG CALEB	BU/UP/2024/3813	MEB
4	OWINO POSH IAN	BU/UP/2024/1067	WAR
5	AHEREZA FAITH	BU/UG/2024/2673	APE
6	MUWUBYA WYCLIFFE	BU/UP/2024/1045	WAR
7	NABUKEERA ANNET RIONAH	BU/UG/2024/2676	AMI
8	AKELLO BARBRA	BU/UP/2024/1001	WAR
9	ATATI EDWINE	BU/UP/2024/3730	AMI
10	OPOKA VINCENT	BU/UG/2024/2675	AMI

ACKNOWLEDGEMENT

First and foremost, we would like to thank the Almighty God for giving us the strength to carry on with our research in group 17. We appreciate all those who contributed to the success of this assignment

The willingness of each one of us to invest time and provide constructive feedback has been immensely valuable in this assignment. Finally, we would like to express our sincere gratitude to all the sources and references that have been mentioned in this report.

ABSTRACT

We started our research from 26th, September, 2025 in the university library where we were exposed to different numerical methods in the various sections through Group 17, as we obtained research on the algorithm's development, control structures ad modules 5.1 for solving mathematical problems such as Euler, notch Kunta, Secant and Newton Raphson, which then helped us to plot graphs that compare the problems analytical solutions to the solutions obtained by different methods. The information we used was obtained from MATLAB textbooks and YOU TUBE tutorials. Plotting the time, accuracy graphs and the number of steps followed graph. We divided group members accordingly to achieve this tasks, different individuals were assigned different numerical methods in order to ease the work and to save time.

DEDICATION

We dedicate this report to all the individuals especially Group 17 members, who sacrificed their time and energy in the process of formulating and producing this report. To our lecturer Mr. Maseruka Benedicto whose guidance and expertise have been of great help, your mentorship and motivated feedback have shaped our understanding.

DECLARATION

We hereby certify and confirm that the information in this report is out of our own efforts, research and it has never been submitted in any institution for any academic award

	NAME	REG NUMBER	PROGRAM	
1	AHAISIBWE CHRISTOPHER	BU/UP/2024/0824	AMI	
2	GIFT EMMANUEL	BU/X/2024/3250	WAR	
3	ADWONG CALEB	BU/UP/2024/3813	MEB	
4	OWINO POSH IAN	BU/UP/2024/1067	WAR	
5	AHEREZA FAITH	BU/UG/2024/2673	APE	
6	MUWUBYA WYCLIFFE	BU/UP/2024/1045	WAR	
7	NABUKEERA ANNET RIONAH	BU/UG/2024/2676	AMI	
8	AKELLO BARBRA	BU/UP/2024/1001	WAR	
9	ATATI EDWINE	BU/UP/2024/3730	AMI	
10	OPOKA VINCENT	BU/UG/2024/2675	AMI	

APPROVAL

We are presenting this report which was successfully written and produced under our efforts as Group 17 giving details of the assignment carried out.

Name…….

Date…….....

Signature…….

TABLE OF CONTENTS

ACKNOWLEDGEMENT ii
ABSTRACT iii
DEDICATION iv
DECLARATION v
APPROVAL vi
CHAPTER ONE 1
1.1 Background. 1
1.2 Historical Development 1
CHAPTER 2: STUDY METHODOLOGY 2
2.1 Introduction 2
2.2 Root Finding Methods for Financial Modeling 2
2.3 Newton-Raphson Method 3
2.4 Secant Method 3
2.5 Visualization - Root Finding Comparison 5
2.6 Real-world problem: Population growth with carrying capacity 10
2.7 Euler's Method 10
2.8 Runge-Kutta 10
2.9 Plotting Results for Differential Equations 12
CHAPTER 3 18
3.1 Conclusion and Learning Experience 18
3.2 References and Resources 18

CHAPTER ONE

INTRODUCTION

1.1 Background.

MATLAB, which stands for matrix laboratory, is a high-performance programming language and environment designed primarily for technical computing. Its origins trace back to the late 1970s when Cleve Moler, a professor of computer science, developed it to provide his students with easy access to mathematical software libraries without requiring them to learn Fortran.

1.2 Historical Development

- **Initial Development**: The first version of MATLAB was created in Fortran in the late 1970s as a simple interactive matrix calculator. This early iteration included basic matrix operations and was built on top of two significant mathematical libraries: LINPACK and EISPACK, which were developed for numerical linear algebra and eigenvalue problems, respectively.

- **Commercial Launch**: MATLAB was officially launched as a commercial product in
1984 by MathWorks, a company founded by Moler along with Jack Little and Steve Bangert. This marked the transition from a simple calculator to a comprehensive programming environment. The software was reimplemented in C, enhancing its capabilities with the addition of user-defined functions, toolboxes, and graphical interfaces.

- **Expansion and Toolboxes**: Over the years, MATLAB has expanded significantly. By the late 1980s, it had introduced several specialized toolboxes for various applications, including control systems and signal processing. The introduction of the *Simulink* environment further allowed users to model and simulate dynamic systems graphically.

- **Modern Enhancements**: Recent versions of MATLAB have introduced features like the *Live Editor*, which allows users to create interactive documents that combine code, output, and formatted text. This evolution reflects MATLAB's ongoing adaptation to meet the needs of its diverse user base across academia and industry.

CHAPTER 2: STUDY METHODOLOGY

2.1 Introduction

At the start time of the research, we divided ourselves accordingly depending on the numerical methods you talked about. At least each member had to look for information

a. Newton's Raphson method
b. Secant method
c. Euler's method
d. Notch kunta method

2.2 Root Finding Methods for Financial Modeling

```
clear;
clc;


% Problem: Find the interest rate that gives a future value of Ugx15,000,000
% for an investment of Ugx10,000,000 over 5 years with monthly compounding
% Where: FV = Future Value, PV = Present Value, r = interest rate, n =
% years, t = period
% FV = PV * (1 + r/12)^(12*t)


PV = 10000000; FV_target = 15000000; t = 5; n = 12;


% Define the function
f = @(r) PV * (1 + r/n)^(n*t) - FV_target; %function
f_prime = @(r) PV * n*t * (1 + r/n)^(n*t - 1) / n; % derivative
```

2.3 Newton-Raphson Method

```
fprintf('=== NEWTON-RAPHSON METHOD ===\n');
tic;
r0 = 0.05; % Initial value (5%)
tolerance = 1e-8; %accuracy(dps)
max_iter = 100;
errors_newton = zeros(1,max_iter); %array for storing the error outputs


for i = 1:max_iter
    f_val = f(r0);
    f_deriv = f_prime(r0);
    r_new = r0 - f_val/f_deriv;

    error = abs(r_new - r0);
    errors_newton(i) = error;

    if error < tolerance
        break;
    end
    r0 = r_new;
    iter0 = i;
end
```

```
Root: r = 0.081368 (8.1368%)
Iterations: 5, Time: 0.020498 seconds
```

```
time_newton = toc;
fprintf('Root: r = %.6f (%.4f%%)\n', r0, r0*100);
fprintf('Iterations: %d, Time: %.6f seconds\n', iter0, time_newton);
```

2.4 Secant Method

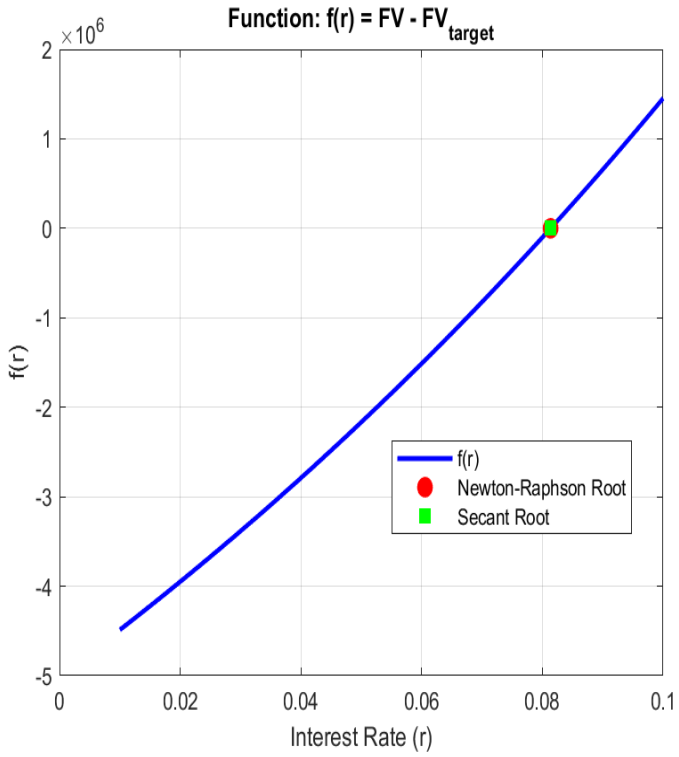
```
fprintf('=== SECANT METHOD ===\n');
tic;
r0_secant = 0.04; % First initial guess
r1_secant = 0.06; % Second initial guess
tolerance = 1e-8; % accuracy (dps)
max_iter = 100;


r_secant = [r0_secant, r1_secant];
errors_secant = [];


for i = 1:max_iter
    f0 = f(r_secant(end-1));
    f1 = f(r_secant(end));
```

```
    if abs(f1 - f0) < eps
        break;
    end

    r_new = r_secant(i+1) - f1 * (r_secant(i+1) - r_secant(i)) / (f1 - f0);
    r_secant(i+2) = r_new;

    error = abs(r_secant(i+2) - r_secant(i+1));
    errors_secant(i:i) = error;

    if error < tolerance
        break;
    end
end


time_secant = toc;
fprintf('Root: r = %.6f (%.4f%%)\n', r_secant(end), r_secant(end)*100);
fprintf('Iterations: %d, Time: %.6f seconds\n', length(r_secant)-2, time_secant);
```

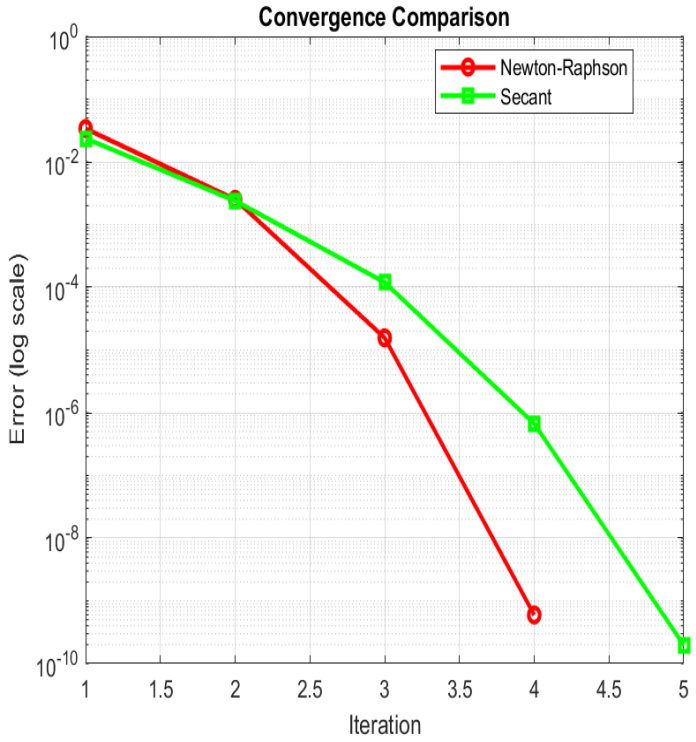
```
Root: r = 0.081368 (8.1368%)
Iterations: 5, Time: 0.020498 seconds
```

2.5 Visualization - Root Finding Comparison

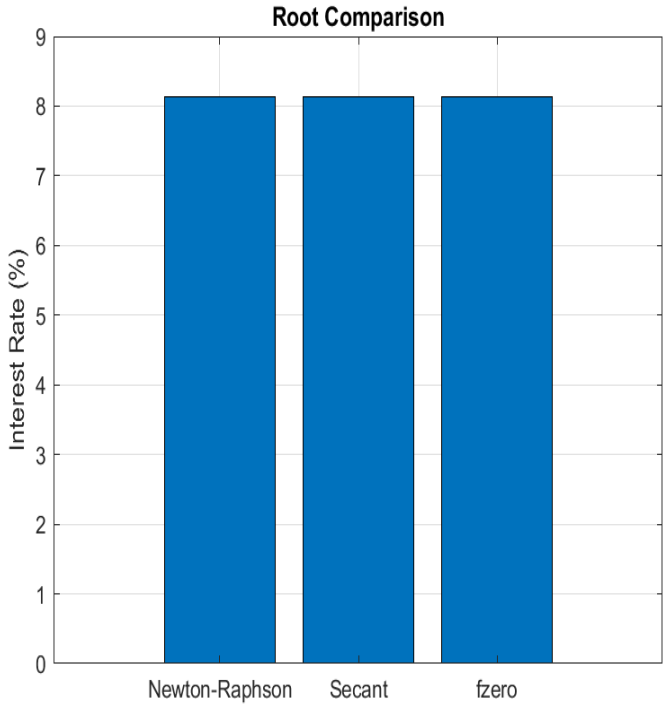
```
figure;
% Function plot
r_range = linspace(0.01, 0.1, 100);
f_values = arrayfun(f, r_range);
plot(r_range, f_values, 'b-', 'LineWidth', 2);
hold on;
plot(r0, f(r0), 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'red');
plot(r_secant(end), f(r_secant(end)), 'gs', 'MarkerSize', 8, 'MarkerFaceColor', 'green');
xlabel('Interest Rate (r)');
ylabel('f(r)');
title('Function: f(r) = FV - FV_{target}');
legend('f(r)', 'Newton-Raphson Root', 'Secant Root', 'Location', 'best');
grid on;
saveas(gcf,'function_plot.png');
```



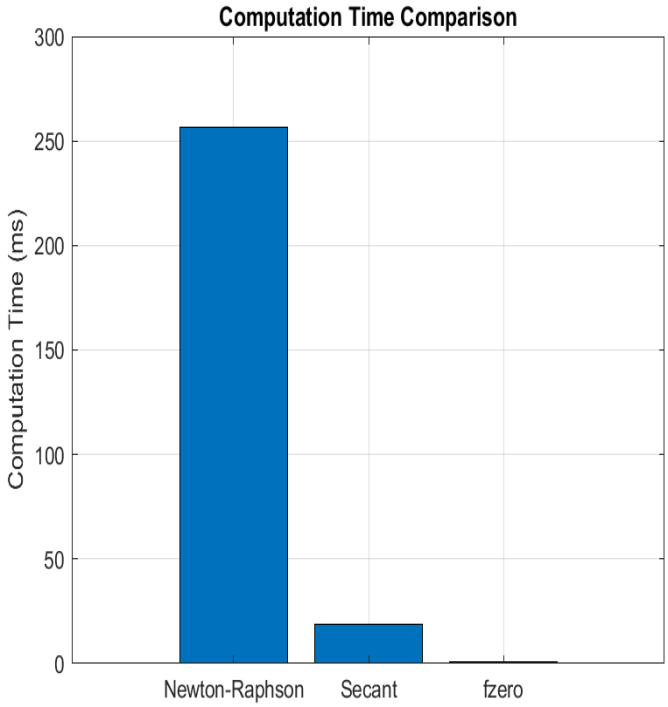
```
% Convergence comparison
figure;
semilogy(1:length(errors_newton), errors_newton, 'r-o', 'LineWidth', 2);
hold on;
semilogy(1:length(errors_secant), errors_secant, 'g-s', 'LineWidth', 2);
xlabel('Iteration');
ylabel('Error (log scale)');
title('Convergence Comparison');
legend('Newton-Raphson', 'Secant', 'Location', 'best');
grid on;
saveas(gcf,'convergence_comparison.png');
```



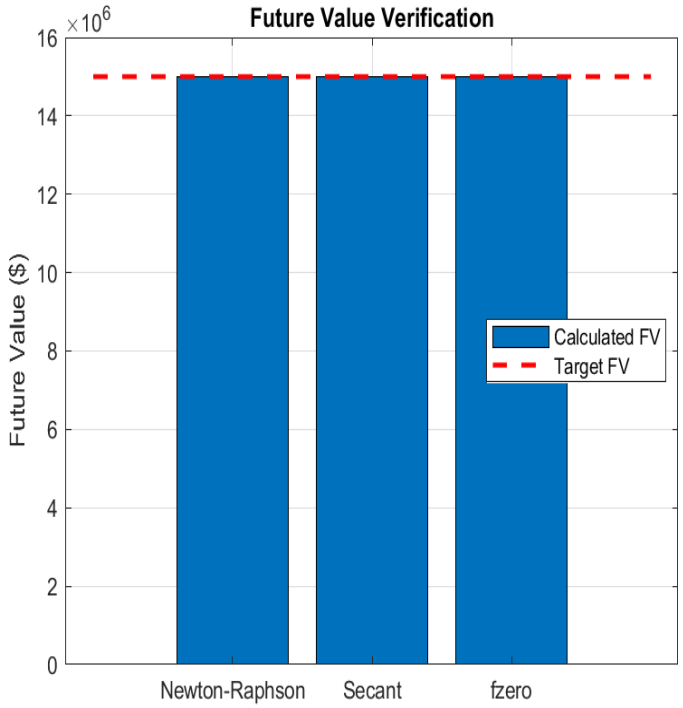
```
% Verification
figure;
r_analytical = fzero(f, 0.05); % MATLAB's built-in function
methods = {'Newton-Raphson', 'Secant', 'fzero'};
roots = [r0, r_secant(end), r_analytical];
times = [time_newton, time_secant, 0.001]; % Approximate fzero time
bar(roots*100);
set(gca, 'XTickLabel', methods);
ylabel('Interest Rate (%)');
title('Root Comparison');
grid on;
saveas(gcf,'Verification.png');
```



```
% Time comparison
figure;
bar(times*1000); % Convert to milliseconds
set(gca, 'XTickLabel', methods);
ylabel('Computation Time (ms)');
title('Computation Time Comparison');
grid on;
saveas(gcf,'Time_comparison.png');
```



```
% Final verification
figure;
FV_newton = PV * (1 + r0/n)^(n*t);
FV_secant = PV * (1 + r_secant(end)/n)^(n*t);
FV_analytical = PV * (1 + r_analytical/n)^(n*t);
bar([FV_newton, FV_secant, FV_analytical]);
hold on;
plot([0, 4], [FV_target, FV_target], 'r--', 'LineWidth', 2);
set(gca, 'XTickLabel', methods);
ylabel('Future Value ($)');
title('Future Value Verification');
legend('Calculated FV', 'Target FV', 'Location', 'best');
grid on;
saveas(gcf,'Final_verification.png');
sgtitle('Root Finding Methods: Compound Interest Rate Calculation', 'FontSize', 14, 'FontWeight', 'bold');
```



2.6 Real-world problem: Population growth with carrying capacity

dP/dt = r*P*(1 - P/K) Where: P = population, r = growth rate, K = carrying capacity

```
% Parameters
r = 0.1;        % Growth rate
K = 1000;       % Carrying capacity
P0 = 100;       % Initial population
t_span = [0, 50]; % Time span
h = 0.1;        % Step size


% Analytical solution
P_analytical = @(t) K ./ (1 + ((K - P0) / P0) * exp(-r * t));
```

2.7 Euler's Method

```
fprintf('=== EULER"S METHOD ===\n');
tic;
t_euler = t_span(1):h:t_span(2);
P_euler = zeros(size(t_euler));
P_euler(1) = P0;


for i = 1:length(t_euler)-1
    dPdt = r * P_euler(i) * (1 - P_euler(i)/K);
    P_euler(i+1) = P_euler(i) + h * dPdt;
end
time_euler = toc;
```

2.8 Runge-Kutta

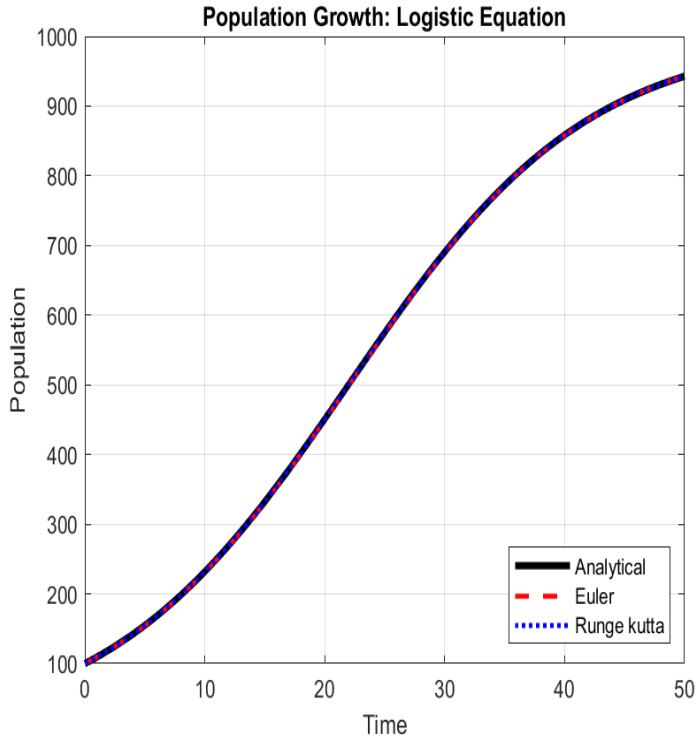
```
fprintf('=== RUNGE-KUTTA  ===\n');
tic;
t_rk = t_span(1):h:t_span(2);
P_rk = zeros(size(t_rk));
P_rk(1) = P0;


for i = 1:length(t_rk)-1
    k1 = r * P_rk(i) * (1 - P_rk(i)/K);
    k2 = r * (P_rk(i) + 0.5*h*k1) * (1 - (P_rk(i) + 0.5*h*k1)/K);
    k3 = r * (P_rk(i) + 0.5*h*k2) * (1 - (P_rk(i) + 0.5*h*k2)/K);
    k4 = r * (P_rk(i) + h*k3) * (1 - (P_rk(i) + h*k3)/K);
```

```
    P_rk(i+1) = P_rk(i) + (h/6) * (k1 + 2*k2 + 2*k3 + k4);
end
time_rk4 = toc;


% Error Analysis
P_analytical_values = P_analytical(t_euler);


error_euler = abs(P_euler - P_analytical_values);
error_rk = abs(P_rk(1:length(t_euler)) - P_analytical_values);


fprintf('=== ERROR ANALYSIS ===\n');
fprintf('Maximum Error - Euler: %.6f\n', max(error_euler));
fprintf('Maximum Error - RK4: %.6f\n', max(error_rk));
fprintf('Computation Time - Euler: %.6f seconds\n', time_euler);
fprintf('Computation Time - Runge kutta: %.6f seconds\n', time_rk4);
```

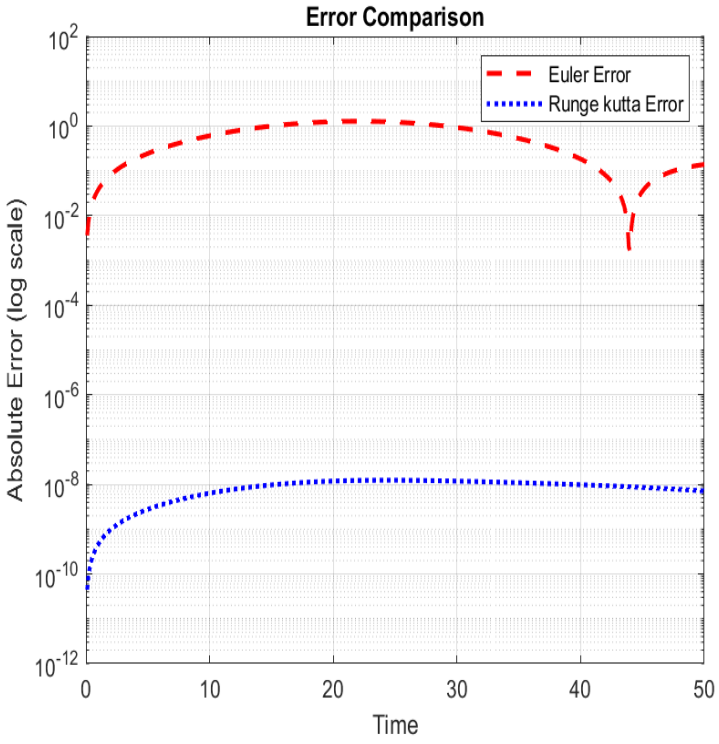
```
Maximum Error - Euler: 1.277479
Maximum Error - RK4: 0.000000
Computation Time - Euler: 0.004431 seconds
Computation Time - RK4: 0.006430 seconds
```

2.9 Plotting Results for Differential Equations

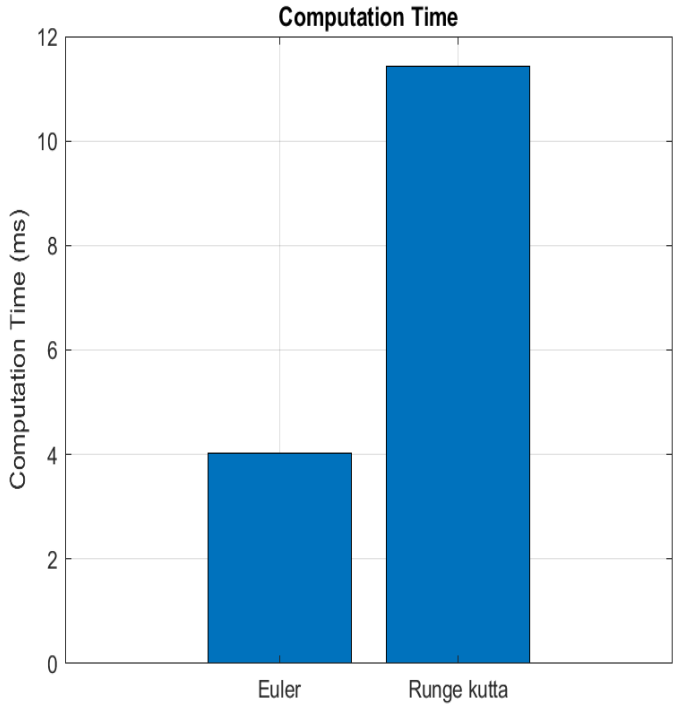
```
% Population growth comparison
figure;
t_dense = linspace(t_span(1), t_span(2), 1000);
plot(t_dense, P_analytical(t_dense), 'k-', 'LineWidth', 3, 'DisplayName', 'Analytical');
hold on;
plot(t_euler, P_euler, 'r--', 'LineWidth', 2, 'DisplayName', 'Euler');
plot(t_rk, P_rk, 'b:', 'LineWidth', 2, 'DisplayName', 'Runge kutta');
xlabel('Time');
ylabel('Population');
title('Population Growth: Logistic Equation');
legend('Location', 'southeast');
grid on;
saveas(gcf,'population_growth_comparison.png');
```



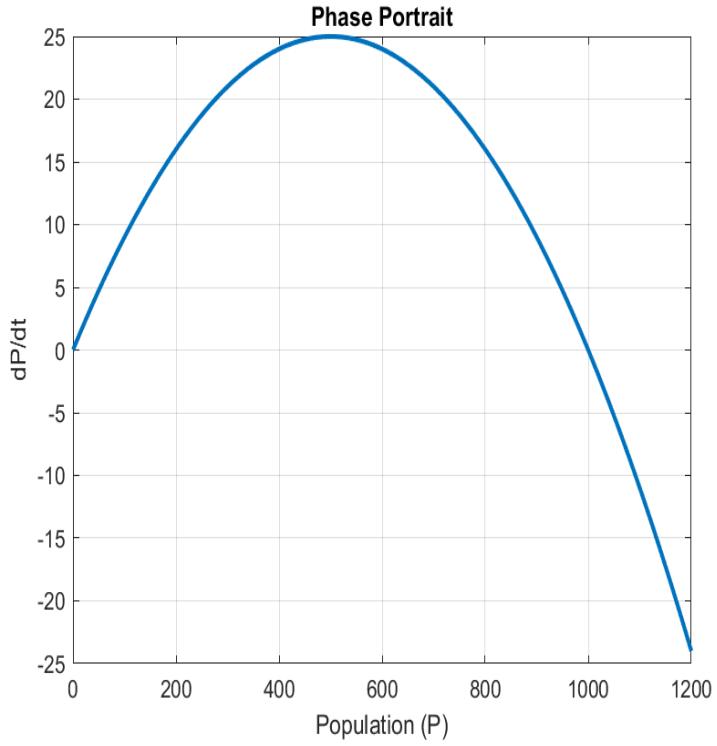
```
% Error comparison
figure;
semilogy(t_euler, error_euler, 'r--', 'LineWidth', 2, 'DisplayName', 'Euler Error');
hold on;
semilogy(t_euler, error_rk, 'b:', 'LineWidth', 2, 'DisplayName', 'Runge kutta Error');
xlabel('Time');
ylabel('Absolute Error (log scale)');
title('Error Comparison');
legend('Location', 'northeast');


grid on;
saveas(gcf,'error_comparison.png');
```



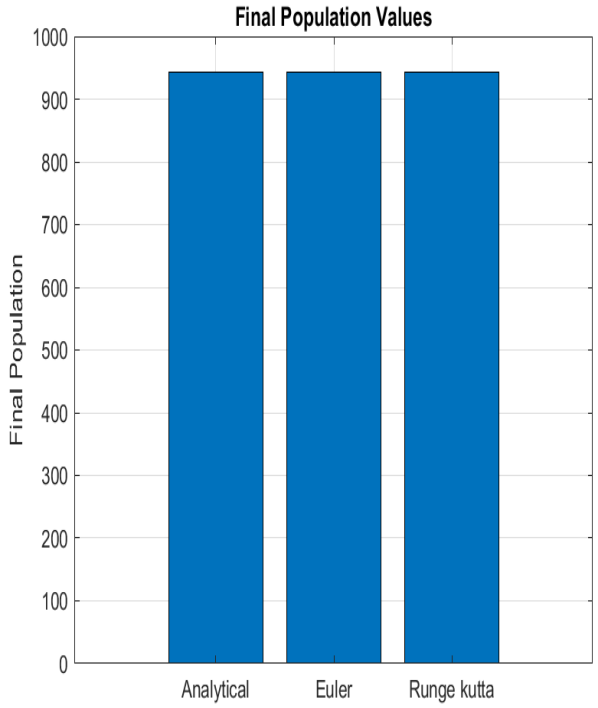
```
% Computation time
figure;
methods_de = {'Euler', 'Runge kutta'};
times_de = [time_euler, time_rk4] * 1000;
bar(times_de);
ylabel('Computation Time (ms)');
title('Computation Time');
set(gca, 'XTickLabel', methods_de);
grid on;
saveas(gcf,'computation_time.png');
```



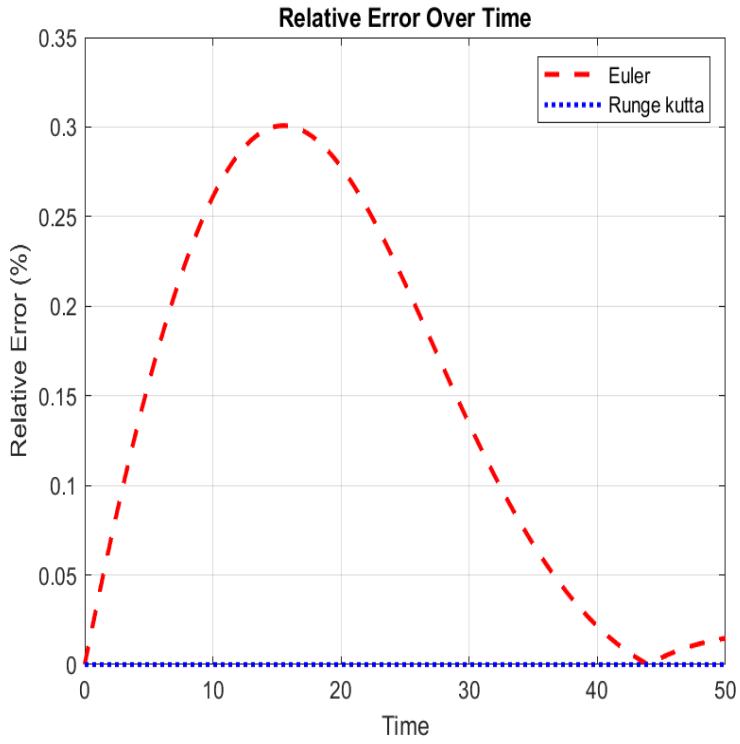
```
% Phase portrait (dP/dt vs P)
figure;
P_range = linspace(0, 1200, 100);
dPdt_range = r * P_range .* (1 - P_range/K);
plot(P_range, dPdt_range, 'LineWidth', 2);
xlabel('Population (P)');
ylabel('dP/dt');
title('Phase Portrait');
grid on;
saveas(gcf,'phase_potrait.png');
```



```
% Final values comparison
figure;
final_values = [P_analytical(t_span(2)), P_euler(end), P_rk(end)];
bar(final_values);
ylabel('Final Population');
title('Final Population Values');
set(gca, 'XTickLabel', {'Analytical', 'Euler', 'Runge kutta'});
grid on;
saveas(gcf,'Final_comparison.png');
```



```
% Relative error over time
figure;
relative_error_euler = error_euler ./ P_analytical_values;
relative_error_rk = error_rk ./ P_analytical_values;
plot(t_euler, relative_error_euler * 100, 'r--', 'LineWidth', 2);
hold on;
plot(t_euler, relative_error_rk * 100, 'b:', 'LineWidth', 2);
xlabel('Time');
ylabel('Relative Error (%)');
title('Relative Error Over Time');
legend('Euler', 'Runge kutta', 'Location', 'northeast');
grid on;
saveas(gcf,'relative_error.png');
sgtitle('Differential Equation Solvers: Population Growth Model');
```



CHAPTER 3

3.1 Conclusion and Learning Experience

This assignment has helped us to acquire knowledge and experience that helped us understand MATLAB programming concepts and gave us experience with the foundations we had acquired from Modules 5.1.

Generally, this assignment provided a practical foundation in data management and problem-solving within a computing environment.

3.2 References and Resources

- MATLAB Documentation - Used for syntax and function guidance on legend(), fprintf(), and print().
- Computer programming lecture notes.