FACULTY OF ENGINEERING AND TECHONOLOGY

COMPUTER PROGRAMMING

ASSIGNMENT REPORT ON THE KNOWLEDGE OBTAINED IN ALGORITHM DEVELOPMENT, CONTROL STRUCTURED PROGRAMMING AND MODULES 1-4 USING MATLAB

BY GROUP SIX

PRESENTED TO MR. MASERUKA BENEDICTO

DATE OF SUBMISSION;

30th SEPTEMBER,2025.

GROUP SIX MEMBERS

NAME	COURSE	REG NUMBER
OPIO SAM ODWORI	WAR	BU/UP/2024/4454
MUWAZA DEOGRATIUS	PTI	BU/UP/2024/5398
KATUNGUKA EMMANUEL	WAR	BU/UP/2024/1030
JOAN RACHAEL LAGU	WAR	BU/UP/2024/1028
KISAKYE PRISCILA PATIANCE	WAR	BU/UP/2024/1034
AYOO LINDA	AMI	BU/UP/2024/5097
WAMBASA GEOFERY	WAR	BU/UP/2024/3259
NAMANYA PRECIOUS	AMI	BU/UP/2024/5255
NANGOLI DERRIK	AMI	BU/UP/2024/3736
APUDA CECILIA LINDI	WAR	BU/UP/2024/3248

DECLARATION

We group six members declare that all the work embodied in this report is of our own efforts and that it was not duplicated from any.

NAMES	SIGNATURE
WAMBASA GEOFREY	
OPIO SAM ODWORI	
JOAN RACHAEL LAGU	
KISAKYE PRISCILA PATIENCE	
NAMANYA PRECIOUS	
AYOO LINDA	
KATUNGUKA EMMANUEL	
MUWAZA DEOGRATIUS	
APUDA CECILIA LINDI	
NANGOLI DERICK	

ACKNOWLEDGEMENT
We thank the Almighty God for the gift of life, knowledge and constant guidance throughout the course of this assignment. We extend our sincere gratitude to our beloved parents for their support in our academics and also our lecturer Mr. Maseruka Benedicto (HOD) for his continuous efforts in ensuring we learn computer programming thoroughly.

We finally thank our fellow group six members for their cooperation and active participation in this assignment through our group leader Wambasa Geofrey.

APPROVAL.

This is to confirm that this report has been prepared and presented by group six without duplication but research.

Date ……

Signature…….

DEDICATION

We dedicate this report to all group SIX members, who worked tirelessly to ensure that its completed.

METHODOLOGY

In this assignment, we utilized our knowledge of algorithm development, structured programming including the knowledge we obtained from modules 1-4.

We were able to use the Newton-Raphson method, the secant method to find solutions to functions, solve differential equations and also solve real world problems which were obtained in different plots.

Table of Content

CHAPTER ONE
INTRODUCTION

HISTORICAL 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.

MATLAB is built around the concept of matrices, making it particularly effective for linear algebra and matrix manipulation. It provides a vast library of built-in functions for mathematical operations, statistics, optimization, and other specialized tasks.

MATLAB offers powerful tools for creating 2D and 3D plots, enabling users to visualize data effectively. Specialized toolboxes extend MATLAB's capabilities, providing functions tailored for specific applications like signal processing, image processing, control systems, and machine learning.

MATLAB can interface with other programming languages (like C, C++, and Python) and software tools, allowing for flexible integration into larger systems. Its interactive environment features a command window, workspace, and editor, making it accessible for both beginners and advanced users.

Historical Background

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.

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.

STUDY METHODOLOGY.

At the start, each member was given a task of making research about the assignment before our first meeting. The research concepts were obtained through watching tutorials on YouTube, reading the modules and consultations from other continuing students especially those in year three and four.

CHAPTER TWO:

Question

In your different groups, utilize the knowledge of algorithm development, control structures and modules 1-4 on the following problems.

(a) All numerical approximation methods for finding the solutions to a function. This includes but not limited to secant, bisection method…

(b) All methods for solving differential equations numerically and these include Euler formular, Runge- kutta

(c) Ensure to apply (a) and (b) on a practical real-world plot

NB Ensure that the different methods are tested on similar programs (minimum of 2)

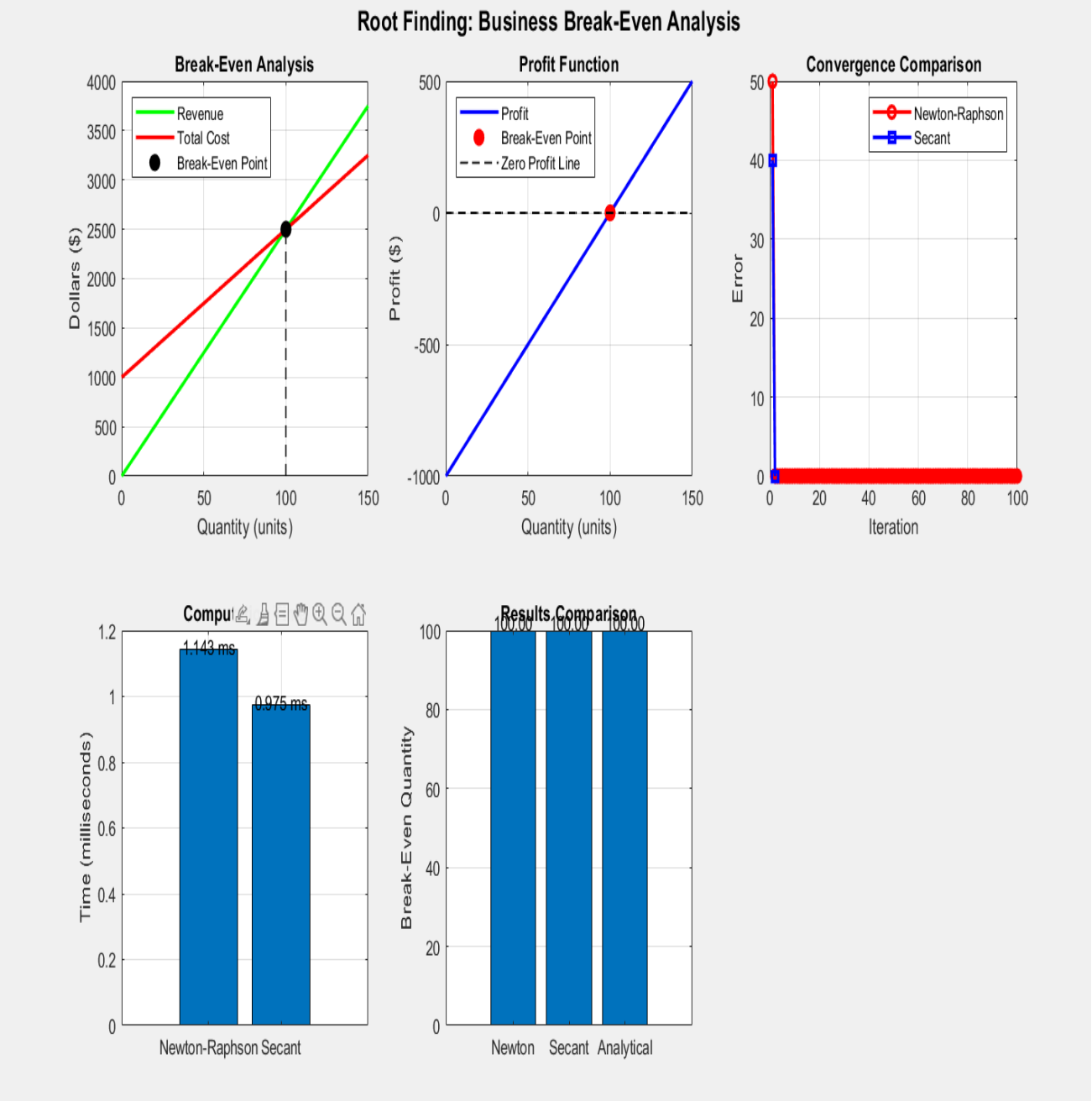
This will help you plot graphs that compare the problem analytical solutions to different methods a long with the computation time.

```
%% Root Finding Methods for Business Break-Even Analysis
clear;
clc;

% Problem: Find break-even point where Revenue = Cost
% Revenue = Price * Quantity
% Cost = Fixed Cost + Variable Cost * Quantity

Price = 25;          % Selling price per unit
Fixed_Cost = 1000;   % Fixed costs
Variable_Cost = 15;  % Variable cost per unit

% Break-even: Price*Q = Fixed_Cost + Variable_Cost*Q
% => (Price - Variable_Cost)*Q - Fixed_Cost = 0
f = @(Q) (Price - Variable_Cost) * Q - Fixed_Cost;
f_prime = @(Q) Price - Variable_Cost; % Constant derivative

%% Newton-Raphson Method
fprintf('=== NEWTON-RAPHSON METHOD ===\n');
fprintf('Business Break-Even Analysis\n');
fprintf('Price: $%.0f, Fixed Cost: $%.0f, Variable Cost: $%.0f\n\n', Price,
Fixed_Cost, Variable_Cost);

tic;
Q0 = 50; % Initial guess
tolerance = 1e-6;
max_iter = 100;

Q_newton = Q0;
iter_newton = 0;
errors_newton = zeros(1,max_iter);

for i = 1:max_iter
```

```
    f_val = f(Q_newton);
    f_deriv = f_prime(Q_newton);
    Q_new = Q_newton - f_val/f_deriv;

    error = abs(Q_new - Q_newton);
    errors_newton(i) = error;

    if error < tolerance
        break;
    end
    Q_newton = Q_new;
    iter_newton = i;
end

time_newton = toc;

% Calculate profit at break-even
Revenue = Price * Q_newton;
Total_Cost = Fixed_Cost + Variable_Cost * Q_newton;
Profit = Revenue - Total_Cost;

fprintf('Break-even quantity: %.2f units\n', Q_newton);
fprintf('Revenue at break-even: $%.2f\n', Revenue);
fprintf('Total Cost at break-even: $%.2f\n', Total_Cost);
fprintf('Profit at break-even: $%.2f\n', Profit);
fprintf('Iterations: %d, Time: %.6f seconds\n', iter_newton, time_newton);

%% Secant Method
fprintf('\n=== SECANT METHOD ===\n');
tic;
Q0_secant = 40;  % First guess
Q1_secant = 60;  % Second guess
tolerance = 1e-6;
max_iter = 100;

Q_secant = [Q0_secant, Q1_secant];
errors_secant = [];

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

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

    Q_new = Q_secant(i+1) - f1 * (Q_secant(i+1) - Q_secant(i)) / (f1 - f0);
    Q_secant(i+2) = Q_new;

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

    if error < tolerance
        break;
    end
end

time_secant = toc;
```

```
% Calculate for secant method
Revenue_secant = Price * Q_secant(end);
Total_Cost_secant = Fixed_Cost + Variable_Cost * Q_secant(end);

fprintf('Break-even quantity: %.2f units\n', Q_secant(end));
fprintf('Revenue at break-even: $%.2f\n', Revenue_secant);
fprintf('Total Cost at break-even: $%.2f\n', Total_Cost_secant);
fprintf('Iterations: %d, Time: %.6f seconds\n', length(Q_secant)-2, time_secant);

%% Analytical solution
Q_analytical = Fixed_Cost / (Price - Variable_Cost);
fprintf('\n=== ANALYTICAL SOLUTION ===\n');
fprintf('Break-even quantity: %.2f units\n', Q_analytical);

%% Visualization - Break-Even Analysis
figure('Position', [100, 100, 1200, 800]);

% Subplot 1: Revenue and Cost curves
subplot(2,3,1);
Q_range = 0:150;
Revenue_curve = Price * Q_range;
Cost_curve = Fixed_Cost + Variable_Cost * Q_range;

plot(Q_range, Revenue_curve, 'g-', 'LineWidth', 2);
hold on;
plot(Q_range, Cost_curve, 'r-', 'LineWidth', 2);
plot(Q_newton, Revenue, 'ko', 'MarkerSize', 8, 'MarkerFaceColor', 'black');
plot([Q_newton, Q_newton], [0, Revenue], 'k--', 'LineWidth', 1);
xlabel('Quantity (units)');
ylabel('Dollars ($)');
title('Break-Even Analysis');
legend('Revenue', 'Total Cost', 'Break-Even Point', 'Location', 'northwest');
grid on;

% Subplot 2: Profit function
subplot(2,3,2);
Profit_curve = Revenue_curve - Cost_curve;
plot(Q_range, Profit_curve, 'b-', 'LineWidth', 2);
hold on;
plot(Q_newton, 0, 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'red');
plot(Q_range, zeros(size(Q_range)), 'k--', 'LineWidth', 1);
xlabel('Quantity (units)');
ylabel('Profit ($)');
title('Profit Function');
legend('Profit', 'Break-Even Point', 'Zero Profit Line', 'Location', 'northwest');
grid on;

% Subplot 3: Convergence comparison
subplot(2,3,3);
plot(1:length(errors_newton), errors_newton, 'r-o', 'LineWidth', 2);
hold on;
plot(1:length(errors_secant), errors_secant, 'b-s', 'LineWidth', 2);
xlabel('Iteration');
ylabel('Error');
title('Convergence Comparison');
legend('Newton-Raphson', 'Secant', 'Location', 'northeast');
grid on;

% Subplot 4: Computation time
```

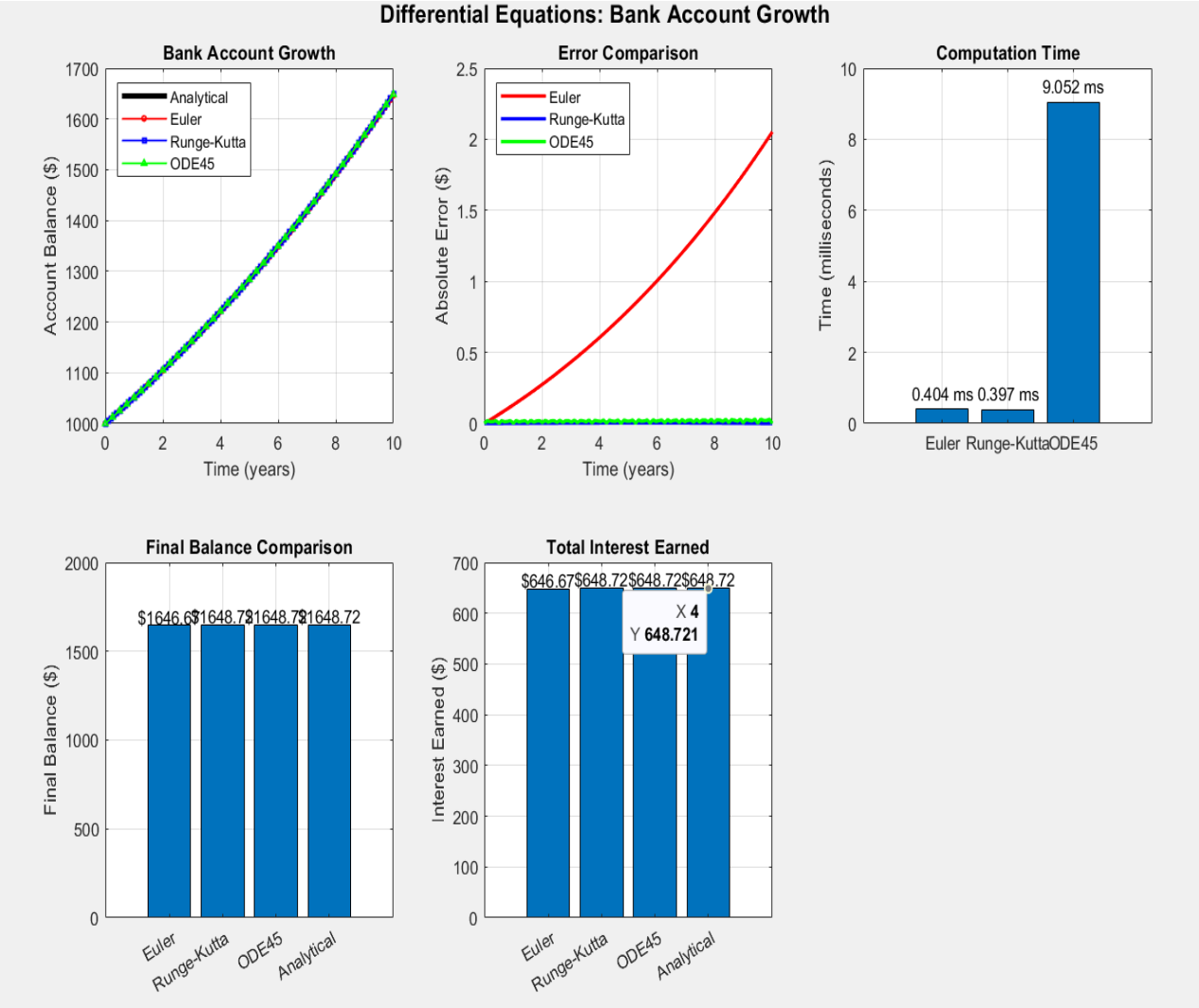
```
subplot(2,3,4);
methods = {'Newton-Raphson', 'Secant'};
times = [time_newton, time_secant] * 1000;

bar(times);
set(gca, 'XTickLabel', methods);
ylabel('Time (milliseconds)');
title('Computation Time');
grid on;

% Add labels on bars
for i = 1:length(times)
    text(i, times(i) + 0.01, sprintf('%.3f ms', times(i)), ...
        'HorizontalAlignment', 'center');
end

% Subplot 5: Results comparison
subplot(2,3,5);
quantities = [Q_newton, Q_secant(end), Q_analytical];
methods_all = {'Newton', 'Secant', 'Analytical'};

bar(quantities);
set(gca, 'XTickLabel', methods_all);
ylabel('Break-Even Quantity');
title('Results Comparison');
grid on;

% Add labels on bars
for i = 1:length(quantities)
    text(i, quantities(i) + 2, sprintf('%.2f', quantities(i)), ...
        'HorizontalAlignment', 'center');
end

sgtitle('Root Finding: Business Break-Even Analysis', 'FontSize', 14,
'FontWeight', 'bold');
```


Root Finding: Business Break-Even Analysis
Break-Even Analysis
Revenue
Total Cost
Break-Even Point
Dollars ($)
Quantity (units)
Profit Function
Profit
Break-Even Point
Zero Profit Line
Profit ($)
Quantity (units)
Convergence Comparison
Newton-Raphson
Secant
Error
Iteration
Compu
1.143 ms
0.975 ms
Time (milliseconds)
Newton-Raphson Secant
Results Comparison
100.00
100.00
100.00
Break-Even Quantity
Newton Secant Analytical

```
% Problem: Bank account with continuous compounding
% dA/dt = r*A
% Where: A = account balance, r = interest rate

r = 0.05;       % Annual interest rate (5%)
A0 = 1000;      % Initial balance ($)
t_span = [0, 10]; % Time span (years)

% Analytical solution: A(t) = A0 * exp(r*t)
A_analytical = @(t) A0 * exp(r*t);

%% Euler's Method
fprintf('=== EULER''S METHOD ===\n');
fprintf('Bank Account with Compound Interest\n');
fprintf('Initial: $%.0f, Rate: %.1f%%, Years: %.0f\n\n', A0, r*100, t_span(2));

tic;
h = 0.1; % Step size (years)
t_euler = t_span(1):h:t_span(2);
A_euler = zeros(size(t_euler));
A_euler(1) = A0;

for i = 1:length(t_euler)-1
    dAdt = r * A_euler(i);
    A_euler(i+1) = A_euler(i) + h * dAdt;
end

time_euler = toc;
final_balance_euler = A_euler(end);
fprintf('Final balance: $%.2f\n', final_balance_euler);
fprintf('Computation time: %.6f seconds\n', time_euler);

%% Runge-Kutta Method
fprintf('=== RUNGE-KUTTA ===\n');
tic;
h_rk = 0.1; % Step size
tt_rk = t_span(1):h_rk:t_span(2);
At_rk = zeros(size(tt_rk));
At_rk(1) = A0;

for i = 1:length(tt_rk)-1
    k1 = r * At_rk(i);
    k2 = r * (At_rk(i) + 0.5*h_rk*k1);
    k3 = r * (At_rk(i) + 0.5*h_rk*k2);
    k4 = r * (At_rk(i) + h_rk*k3);

    At_rk(i+1) = At_rk(i) + (h_rk/6) * (k1 + 2*k2 + 2*k3 + k4);
end

timet_rk = toc;
final_balancet_rk = At_rk(end);
fprintf('Final balance: $%.2f\n', final_balancet_rk);
fprintf('Computation time: %.6f seconds\n', timet_rk);

%% MATLAB's ode45
fprintf('=== MATLAB ODE45 ===\n');
tic;
dAdt_func = @(t, A) r * A;
[t_ode45, A_ode45] = ode45(dAdt_func, t_span, A0);
```

```
time_ode45 = toc;
final_balance_ode45 = A_ode45(end);
fprintf('Final balance: $%.2f\n', final_balance_ode45);
fprintf('Computation time: %.6f seconds\n', time_ode45);

%% Analytical solution
final_balance_analytical = A_analytical(t_span(2));
fprintf('\n=== ANALYTICAL SOLUTION ===\n');
fprintf('Final balance: $%.2f\n', final_balance_analytical);

%% Visualization - Bank Account Growth
figure('Position', [100, 100, 1200, 800]);

% Subplot 1: Account balance over time
subplot(2,3,1);
t_analytical_plot = linspace(t_span(1), t_span(2), 100);
A_analytical_plot = A_analytical(t_analytical_plot);

plot(t_analytical_plot, A_analytical_plot, 'k-', 'LineWidth', 3, ...
    'DisplayName', 'Analytical');
hold on;
plot(t_euler, A_euler, 'ro-', 'LineWidth', 1, 'MarkerSize', 3, ...
    'DisplayName', 'Euler');
plot(tt_rk, At_rk, 'bs-', 'LineWidth', 1, 'MarkerSize', 3, ...
    'DisplayName', 'Runge-Kutta');
plot(t_ode45, A_ode45, 'g^-', 'LineWidth', 1, 'MarkerSize', 3, ...
    'DisplayName', 'ODE45');
xlabel('Time (years)');
ylabel('Account Balance ($)');
title('Bank Account Growth');
legend('Location', 'northwest');
grid on;

% Subplot 2: Error comparison
subplot(2,3,2);
A_euler_interp = interp1(t_euler, A_euler, t_analytical_plot);
At_rk_interp = interp1(tt_rk, At_rk, t_analytical_plot);
A_ode45_interp = interp1(t_ode45, A_ode45, t_analytical_plot);

error_euler = abs(A_euler_interp - A_analytical_plot);
errort_rk = abs(At_rk_interp - A_analytical_plot);
error_ode45 = abs(A_ode45_interp - A_analytical_plot);

plot(t_analytical_plot, error_euler, 'r-', 'LineWidth', 2);
hold on;
plot(t_analytical_plot, errort_rk, 'b-', 'LineWidth', 2);
plot(t_analytical_plot, error_ode45, 'g-', 'LineWidth', 2);
xlabel('Time (years)');
ylabel('Absolute Error ($)');
title('Error Comparison');
legend('Euler', 'Runge-Kutta', 'ODE45', 'Location', 'northwest');
grid on;

% Subplot 3: Computation time
subplot(2,3,3);
methods = {'Euler', 'Runge-Kutta', 'ODE45'};
times = [time_euler, timet_rk, time_ode45] * 1000;

bar(times);
```

```
set(gca, 'XTickLabel', methods);
ylabel('Time (milliseconds)');
title('Computation Time');
grid on;

for i = 1:length(times)
    text(i, times(i) + max(times)*0.05, sprintf('%.3f ms', times(i)), ...
        'HorizontalAlignment', 'center');
end

% Subplot 4: Final balance comparison
subplot(2,3,4);
final_balances = [final_balance_euler, final_balancet_rk, final_balance_ode45,
final_balance_analytical];
methods_all = {'Euler', 'Runge-Kutta', 'ODE45', 'Analytical'};

bar(final_balances);
set(gca, 'XTickLabel', methods_all);
ylabel('Final Balance ($)');
title('Final Balance Comparison');
grid on;

for i = 1:length(final_balances)
    text(i, final_balances(i) + 50, sprintf('$%.2f', final_balances(i)), ...
        'HorizontalAlignment', 'center');
end

% Subplot 5: Interest earned
subplot(2,3,5);
interest_earned = final_balances - A0;
bar(interest_earned);
set(gca, 'XTickLabel', methods_all);
ylabel('Interest Earned ($)');
title('Total Interest Earned');
grid on;

for i = 1:length(interest_earned)
    text(i, interest_earned(i) + 20, sprintf('$%.2f', interest_earned(i)), ...
        'HorizontalAlignment', 'center');
end

sgtitle('Differential Equations: Bank Account Growth', 'FontSize', 14,
'FontWeight', 'bold');
```


Differential Equations: Bank Account Growth
Bank Account Growth
Analytical
Euler
Runge-Kutta
ODE45
Account Balance ($)
Time (years)
Error Comparison
Euler
Runge-Kutta
ODE45
Absolute Error ($)
Time (years)
Computation Time
9.052 ms
0.404 ms
0.397 ms
Time (milliseconds)
Euler
Runge-Kutta
ODE45
Final Balance Comparison
$1646.67
$1648.72
$1648.72
$1648.72
Final Balance ($)
Euler
Runge-Kutta
ODE45
Analytical
Total Interest Earned
$646.67
$648.72
$648.72
$648.72
X 4
Y 648.721
Interest Earned ($)
Euler
Runge-Kutta
ODE45
Analytical

DESCRIPTION FOR THE ABOVE GRAPHS

Subplot 1

1. BREAK EVEN ANALYSIS GRAPH (TOP LEFT)

This graph plots revenue against total costs. It majorly shows the break-even point where revenue equals to the cost.

2. PROFIT FUNCTION GRAPH (top center)

This graph plots profit against quantity. It majorly includes a zero- profit line and marks the Break-Even point.

3. CONVERGENCE COMPARISON GRAPH (top-right)

The plot illustrates errors against iterations for Newton-Raphson and Secant methods. It demonstrates the root finding method that converges faster.

4. COMPUTATIONAL TIME BARCHART (bottom-right)

It compares the time taken by the Newton-Raphson against the Secant method.

5. RESULTS COMPARISON BARCHART (bottom-right)

This compares break-even quantity results between methods, basically showing how they converge to the same solution.

Subplot 2

1. BANK ACCOUNT GROWTH (top left)

This plot can also be referred to as the Time against Balance plot. It shows the account growth curve for different numerical methods for example Euler's method, Runge-Kutta.

Note: all methods generally follow the exponential growth but Euler's method drifts slightly from the exact solution.

2. ERROR COMPARISON PLOT (top-middle)

This is also known as Absolute Error against Time plot.

This majorly highlights why Euler is less accurate for stiff or long-term integration.

3. COMPUTATION TIME (top-right)

It is the most computationally expensive but far more accurate and reliable for real problems.

4. FINAL BALANCE COMPARISON (bottom-left)

For this comparison plot, all methods give nearly the same final balance. Though differences are very small, Euler's method is slightly off.

5. TOTAL INTEREST EARNED (bottom-right)

All comparisons between methods are very close but Euler's method slightly underestimates compared to the analytical method.

CHAPTER THREE:

3.1 CHALLENGES

- Limited time given for the assignment to be completed.
- Referencing errors at times made the work hectic
- Lack of concentration due to the different course units being handled at the same time

3.2 RECOMMENDATIONS

- We recommend that the lecturer to always give us ample time to complete the assignment.

3.3 CONCLUSION AND LEARNING EXPERIENCE

Upon assignment completion, we really appreciated the MATLAB especially from the introduction part to the coverage. This embedded a real-life application of the software into the different engineering aspects. We gained a deeper rhythm on algorithm development, structured programming and also the previous modules 1-4. This experience was of utmost importance to all of us.