FACULTY OF ENGINEERING AND TECHONOLOGY

COMPUTER PROGRAMMING

ASSIGNMENT REPORT ON THE KNOWLEDGE OBTAINED FROM CONCEPTS OF CLASSES, INHERITANCE AND POLYMORPHISM FOR SOLVING COMPUTATIONAL PROBLEMS IN LINE WITH KNOWLEDGE FROM MODULES 3 AND 5 USING MATLAB

BY GROUP SIX

PRESENTED TO MR. MASERUKA BENEDICTO

DATE OF SUBMISSION;

28th, OCTOBER,2025.

AIM: For Partial fulfilment of continuous coursework assessments.

GROUP SIX MEMBERS

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

DECLARATION

We hereby declare that all the work covered in this continuous assessment report is of our own efforts and it has not been duplicated from any source.

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

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 at view of 360*.

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 team without duplication but through 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 numerical methods while implementing the concepts of encapsulation, inheritance and polymorphism with sub classes like integral and differential problems in computational simplicities.

We were able to use the Newton-Raphson method, the secant method to find solutions to functions in reference recursion and dynamic programming, solve differential equations and also solve real world problems.

Table of Content S

CHAPTER ONE

INTRODUCTION

1.1-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.

1.2-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.

1.3-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 taught and others obtained through watching tutorials on YouTube, reading the modules 3 and 5 consultations from other continuing students especially those in year three and four.

CHAPTER TWO:

Question

Whilst implementing the concept of classes, encapsulation, inheritance, polymorphism and abstract classes.

(a)Develop and test a high end backend implementation of a numerical methods application for solving computational problems for simplicity. Apply the codes developed in the previous assignment and ensure the parent class holds all abstract methods with two sub classes one for differential problems and other for integral problems.

2.1-NUMERICAL METHODS

```
classdef (Abstract) NumericalMethod
    % NumericalMethod - Abstract base class for numerical methods
    % Demonstrates abstraction, encapsulation, and polymorphism

    properties (Access = protected)
        % Protected property: accessible only within this class and subclasses
        problemDescription
    end

    methods
        function obj = NumericalMethod(description)
            % Constructor: sets the problem description
            obj.problemDescription = description;
        end

        function displayDescription(obj)
            % Public method to display the problem description
            fprintf('Problem: %s\n', obj.problemDescription);
        end
    end

    methods (Abstract)
        % Abstract method: must be implemented by subclasses
        result = solve(obj)
    end
end
```

2.2-DIFFERENTIALSOLVER

```
classdef DifferentialSolver < NumericalMethod
    % DifferentialSolver - Solves ODEs using Euler's method

    properties (Access = private)
        f       % Function handle for dy/dx
        x0      % Initial x value
        y0      % Initial y value
        h       % Step size
        n       % Number of steps
    end
```

```
    methods
        function obj = DifferentialSolver(description, f, x0, y0, h, n)
            % Call parent constructor
            obj@NumericalMethod(description);
            % Store parameters privately
            obj.f = f;
            obj.x0 = x0;
            obj.y0 = y0;
            obj.h = h;
            obj.n = n;
        end

        function result = solve(obj)
            % Implements Euler's method for ODEs
            x = obj.x0;
            y = obj.y0;
            for i = 1:obj.n
                y = y + obj.h * obj.f(x, y);
                x = x + obj.h;
            end
            result = y;
        end
    end
end
```

2.3-INTEGRALSOLVER

```
classdef IntegralSolver < NumericalMethod
    % IntegralSolver - Solves definite integrals using the trapezoidal rule

    properties (Access = private)
        f    % Function handle for f(x)
        a    % Lower limit
        b    % Upper limit
        n    % Number of subintervals
    end

    methods
        function obj = IntegralSolver(description, f, a, b, n)
            % Call parent constructor
            obj@NumericalMethod(description);
            % Store parameters privately
            obj.f = f;
            obj.a = a;
            obj.b = b;
            obj.n = n;
        end

        function result = solve(obj)
            % Implements trapezoidal rule
            h = (obj.b - obj.a) / obj.n;
            sumVal = obj.f(obj.a) + obj.f(obj.b);
            for i = 1:obj.n-1
                sumVal = sumVal + 2 * obj.f(obj.a + i*h);
            end
            result = (h / 2) * sumVal;
        end
    end
end
```

2.4-TEST-EXAMPLE

```
% Test script for Numerical Methods OOP implementation

% Differential equation example: dy/dx = x + y, y(0) = 1
diffSolver = DifferentialSolver( ...
    'Solve dy/dx = x + y using Euler''s method', ...
    @(x, y) x + y, 0, 1, 0.1, 10);

% Integral example: ∫_0^1 e^x dx
intSolver = IntegralSolver( ...
    'Integrate e^x from 0 to 1 using trapezoidal rule', ...
    @(x) exp(x), 0, 1, 100);

% Display and solve differential problem
diffSolver.displayDescription();
diffResult = diffSolver.solve();
fprintf('Differential equation result: %.6f\n', diffResult);

% Display and solve integral problem
intSolver.displayDescription();
intResult = intSolver.solve();
fprintf('Integral result: %.6f\n', intResult);
```

CHAPTER THREE

3.1-CHALLENGES FACED

- 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 amidst others assignments.

3.3 -CONCLUSION AND LEARNING EXPERIENCE

sUpon assignment completion, we really appreciated the MATLAB especially module 5 and 3. This embedded a real-life application of the software into the different engineering aspects. We gained a deeper rhythm on inheritances polymorphism and encapsulation in conjunction with numerical computation programming methods. This experience was of great importance to all of us.