International Journal of Mathematics and Statistics Invention (IJMSI)
E-ISSN: 2321- 4767 P-ISSN: 2321-4759
www.ijmsi.org Volume 8 Issue 9 || September, 2020 || PP 20-34

On Desktop Application for Some Selected Methods in Ordinary Differential Equations and Optimization Techniques

[1]Buah, A. M.,[2]Alhassan, S. B. and[3]Boahene,B. M.
[1] Department of Mathematical Sciences, University of Mines and Technology, Tarkwa, Ghana
[2] Department of Mathematical Sciences, University of Mines and Technology, Ghana
[3] Department of Mathematical Sciences, University of Mines and Technology, Ghana

ABSTRACT*-Numerical methods are algorithms used to give approximate solutions to mathematical problems. For many years, problems solved numerically were done manually, until the software was developed to solve such problems. The problem with some of this softwareis that, though some are free, the environment they offer is not user friendly. With others, it would cost you a fortune to get access to them and some suffer execution time and overall speed. In this research, a desktop application with a friendly user interface, for solving some selected methods in numerical methods for Ordinary Differential Equations (ODEs) and sequential search methods in optimization techniques has been developed using Visual C++. The application can determine the error associated with some of the methods in ODEs if the user supplies the software with the exact solution. The application can also solve optimization problems (minimization problems) and return the minimum function value and the point at which it occurs.*

KEYWORDS:*ODEs, optimization, desktop application*

Date of Submission: 11-11-2020 Date of Acceptance: 27-11-2020

I. INTRODUCTION

A Differential Equation (DE) is an equation involving a function and its derivatives [1]. Differential equations are called Partial Differential Equations (PDEs) or Ordinary Differential Equations (ODEs) according to whether or not they contain partial derivatives. Often, many systems described by these equations are so complex that a purely analytical solution to the equation is not manageable. In such cases, numerical techniques are employed to obtain approximate solutions.Optimization techniques deal with finding the best solution to a given problem from a set of feasible solutions. The concept of optimization (minimization or maximization) is fundamental in life; what is desirable, what brings pleasure and comfort is maximized and what is undesirable is minimized. ODEs and Optimization techniques are applied in solving real-life problems in engineering, operations research, and economics.

Most of the techniques for solving optimization problems and ODEs based on numerical approximations were developed before programmable computers existed [2]. As programmable computers have increased in speed and decreased in cost, complex systems of ODEs and optimization techniques can be solved with simple programs written to run on a common PC. Many such programs exist. For instance, SAS, ADMB, CUTEr, Scilab, and CPLEX are used to solve problems in optimization techniques and COPASI, MATLAB, GNU Octave, and Sage Math for solving ODEs. The problem with some of these applications is that, though they are free, the environment they offer is not user friendly. With others, it would cost you a fortune to get access to them and some suffer execution time and overall speed.This project seeks to offer a free program for solving ODEs and optimization problems with a friendly Graphical User Interface (GUI), faster execution time, and overall speed.

II. PRELIMINARIES

Numerical methods are laid down rules used to solve mathematical problems. This involves only the operation of arithmetic [4].These rules might sometimes include some kind of test to see whether some conditions have been satisfied and if not the set of rules are applied again.Many problems that suffice in the field of engineering cannot be solved by analytical techniques. The knowledge of numerical methodology is essential for determining approximate solutions to such problems. Numerical methods, in one form or another, have been studied for several centuries. Some examples are Newton's method of approximating the solution of an algebraic or transcendental equation, the Gaussian elimination method for solving linear systems of equations, the Euler, Modified Euler, and Runge-Kutta’s methods for solving initial-value problems.

Since the arrival of computers, the potential of these methods has been realized and the entire chapter of numerical methods has changed. Iterative methods can be used with much greater speed. Large systems of equations can also be solved numerically. Many non-linear equations that were intractable in the past can now be solved approximately by numerical methods. As a result of these possibilities, mathematical modeling is now more realistic and a new field of numerical simulation has been opened up.

2.1 Numerical Methods forOrdinary Differential Equations

A Differential Equation (DE) is an equation involving a function and its derivatives.

Examples are:

(i) $y'(x) = 7x$ (1)

(ii) $y''(x) = 7y'(x) - 6x^2$ (2)

(iii) $\left(\frac{d^2y}{dx^2}\right)^3 + 2\left(\frac{dy}{dx}\right)^5 = 7$ (3)

Differential equations are classified according to the highest derivative which occurs in them. Thus Equation (1) is a first-order differential equation, and Equation (2) and (3) are second-order differential equations. Differential equations are called Partial Differential Equations (PDEs) or Ordinary Differential Equations (ODEs) according to whether or not they contain partial derivatives [5].

Some numerical methods for ODEs include Taylor's method, Euler's Method, Modified Euler's method, Merson's method, Runge-Kutta-Fehlberg method, Runge-Kutta method, Adams-Moulton-Bashforth predictor/corrector method, Shooting method, Picard's method, and Finite deference method. Four of these methods are considered in this work.

2.1.1. Euler's Method

This is the simplest numerical method for solving initial value problems. This method is not an efficient numerical method, but many of the ideas involved in the numerical solutions of differential equations are introduced most simply with it.

Given $y' = f(x,y), h, y_0, x_0$ *and* x_n, Euler's method is given as Equation (4):

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{4}$$

2.1.2. Modified Euler's Method

The source of error in Euler's method is its failure to take into account, the curvature of the solution curve at a point (x_i, y_i) when using the tangent line approximation to the curve to estimate y_{i+1}. An improvement can be obtained by using a two-stage process to arrive at a modified gradient $\tilde{f}(x_i, y_i)$ that can be used in Euler's method in place of $f(x_i, y_i)$. The formula for computing y_{n+1}^p and y_{n+1}^c is given as Equation (5) and (6).

Algorithm:

1. Compute $y_{n+1}^p = y_n + hf(x_n + y_n)$ (5)
2. Compute $y_{n+1}^c = y_n + \frac{h}{2}\{f(x_n, y_n) + f(x_{n+1}, y_{n+1}^p)\}$ (6)

2.1.3. Fourth Order Runge-Kutta Method

The Runge-Kutta's method for solving first-order differential equations is widely used and provides a high degree of accuracy. Again, as with the two previous methods, Runge-Kutta's method is a step-by-step process where results are tabulated for a range of values of x [6]. Although several intermediate calculations are needed at each stage, the method is fairly straightforward.

To solve the differential equation $y'(x) = f(x,y)$, with the Runge-Kutta's method given $y = y_0$, $x = x_0$ and a step size *h*, the following steps are followed.

1. $k_1 = f(x_n, y_n)$ (7)
2. $k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1)$ (8)
3. $k_3 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2)$ (9)
4. $k_4 = (x_n + h, y_n + hk_3)$ (10)

5. Evaluate Equations (7), (8), (9) and (10)
6. Use the values determined from the evaluation to calculate Equation (11):
7. $y_{n+1} = \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4]$

(11)

2.1.4. Adams-Bashforth Predictor-Corrector Method

The Adams-Bashforth predictor-corrector method is one of the best multistep methods which make use of information at the four preceding mesh points $x_n, x_{n-1}, x_{n-2}\ and\ x_{n-3}$to calculate y at the next mesh point x_{n+1}.Before this method can be used, it is necessary to compute $y_1, y_2, and\ y_3$ by some starting method of high accuracy like the Runge-Kutta method.
The predictor formula for the method is given by Equation (12)

$$y_{n+1}^{p} = y_n + \left(\frac{h}{24}\right)[55y_n^{'} - 59y_{n-1}^{'} + 37y_{n-2}^{'} - 9y_{n-3}^{'}]$$

(12)

and the corrector formula is given by Equation (13)

$$y_{n+1}^{c} = y_n + \left(\frac{h}{24}\right)[9y_{n+1}^{'} + 19y_n^{'} - 5y_{n-1}^{'} + y_{n-2}^{'}]$$

(13)

2.2. Numerical Optimization

Optimization is essentially about finding the best solution to a given problem from a set of feasible solutions.

The optimization technique is a very old subject of great interest; examples can be found, searching deep into human history, for example, the need for finding the best way to produce food yielded, finding the best piece of land for producing, as well as the best ways of treatment of the chosen land and the chosen seedlings to get the best results [7].Manufacturers always try to find the best ways of getting maximum income out of the minimum expenses from the very beginning of manufacturing.
In the simplest case, this means solving problems in which one seeks to minimize or maximize a real function by systematically choosing the values of the real or integer variables from within an allowed set.

Methods of numerical optimization include:
Bisection method, Newton's method, Davis-Swapy-Campey (DSC) method, Powel's method, Coggin's method, Extrema condition method, Davidon's cubic search method, Exhaustive search method, Dichotomous search method, Two-point-equal search method, Three-point-equal search method, and Fibonacci search method.
Four of these methods are considered in this project.

2.2.1. Dichotomous Search Method

The Dichotomous Search Method computes the midpoint $\frac{a+b}{2}$, and then moves slightly to either side of the midpoint to compute two test points: $\frac{a+b}{2} \pm \varepsilon$ where ε is a very small number.
The objective is to place the two test points as close together as possible. The procedure continues until it gets within some small interval containing the optimal solution.
Steps:
1. Initialize: Choose a small number ε > 0, such as 0.01. Select a small t such that
0 < t < b – a, called the length of uncertainty for the search. Calculate the number of
Iterations, n, can be calculated using Equation (14)

$$n = \left|\frac{\ln(\frac{b-a}{t})}{ln2}\right| \quad (14)$$

2. Fork = 1 to n, do steps 3 and 4
3. $x_1 = \left(\frac{a+b}{2}\right) - \frac{\varepsilon}{2}andx_2 = \left(\frac{a+b}{2}\right) + \frac{\varepsilon}{2}$
4. If $f(x_1) < f(x_2)$then $b = x_2$ else $a = x_1$, k+=1return to step3.
5. Let $x^* = \frac{a+b}{2}, min = f(x^*)$

2.2.2. Two-Point Equal Search Method

In this method, $x_1 and\ x_2$ are chosen in such a way that the interval [a, b] is divided into three equal intervals as shown in Fig 1.

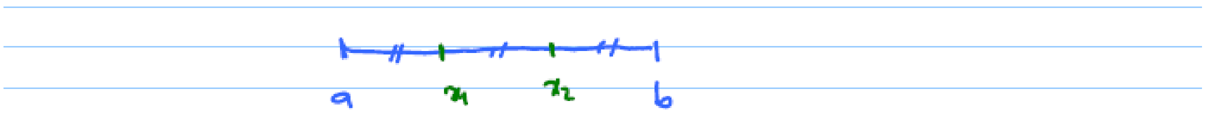


Fig 1 Two-Point Equal Search Method

Steps:

1 Initialize a, b, and t, where t refers to the interval of uncertainty

2 While $|b-a| \geq t$, evaluate Equation (15) and (16)

$$x_1 = a + \left(\frac{b-a}{3}\right) \quad (15)$$

$$x_2 = b - \left(\frac{b-a}{3}\right) \quad (16)$$

Evaluate $f(x_1)$ and $f(x_2)$and if $f(x_1) \leq f(x_2)$, then, a = a and b = x_2 else, a = $x_1$and b = b.

3 If $f(x_1) \leq f(x_2)$then $x^* = x_1$else $x^* = x_2$and $min = f(x^*)$

2.2.3. Three-Point Equal Search Method

The three-point equal search method (interval halving method) involves dividing the interval of uncertainty into four equal parts as demonstrated in Figure 2.

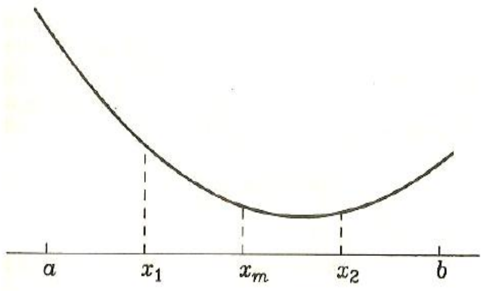


Fig 2 Three-Point Equal Search Method

Figure 2 shows a region in the interval (a, b). Three points divide the search space into four regions. The fundamental rule for region elimination is used to eliminate a portion of the search space based on function values at the three chosen points. $\boldsymbol{x_1}, \boldsymbol{x_m}, \boldsymbol{x_2}$ are three search points. Two of the function values are compared at a time and some region is eliminated.

Algorithm:

1. Choose a lower bound a and an upper bound b and a small value ε for the desired accuracy.

Set $x_m = \frac{a+b}{2}, L_0 = L = b - a.$

2. Set $x_1 = a + \frac{L}{4}, x_2 = b - \frac{L}{4}$. Compute $f(x_1)$ *and* $f(x_2)$.
3. If $f(x_1) < f(x_m)$, set $b = x_m$, $x_m = x_1$. Go to step 5, else go to step 4.
4. If $f(x_2) < f(x_m)$, set $a = x_m$, $x_m = x_2$. Go to step 5, else $a = x_1, b = x_2$ go to step 5.
5. Calculate L = b – a. If |L| < ε, terminate. Else go to step 2.

2.2.4. Fibonacci Search Method

Here, the search interval is reduced according to the Fibonacci numbers $F_n = F_{n-1} + F_{n-2}$.
Where n = 2, 3, 4 ... and $F_0 = F_1 = 1$.

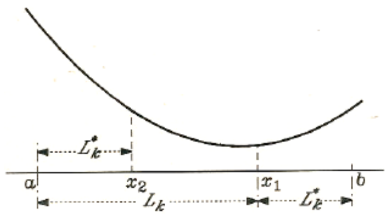


Fig 3 Fibonacci Search Method

Algorithm:
1. Choose a lower bound a and an upper bound b. Set L= b - a. Assume the desired number of function evaluations to be n. Set k = 2
2. Compute $L_i^* = \left(\frac{F_{n-k}}{F_{n-(k-2)}}\right) L$, set $x_1 = a + L_i^* \;\; and \;\; x_2 = b - L_i^*$, i = 1,2,3 … n-1
3. Compute one of $f(x_1)$ or $f(x_2)$ which was not evaluated earlier and use the fundamental region elimination rule to eliminate a region. That is, if $f(x_1) < f(x_2)$, the point to right (L_k^* in Fig 2). Set new a and b.
4. If k = n, terminate the search. Else, set k = k + 1 and go to step 2.

2.3. *Microsoft Visual C++*

Microsoft Visual C++ (often abbreviated to MSVC) is an integrated development environment (IDE) product from Microsoft for the C, C++, and C++/CLI programming languages [8]. MSVC is proprietary software; it was originally a standalone product but later became a part of Visual Studio and made available in both trialware and freeware forms. It features tools for developing and debugging C++ code, especially code written for the Windows API, DirectX, and NET.

Many applications require redistributable Visual C++ runtime library packages to function correctly. These packages are often installed independently of applications, allowing multiple applications to make use of the package while only having to install it once. These Visual C++ redistributable and runtime packages are mostly installed for standard libraries that many applications use.

Microsoft Visual C++ 2012 Redistributable (x64) - 11.0.61030 安装程序

Microsoft Visual C++ 2012
Redistributable (x64) - 11.0.61030

MICROSOFT软件许可条款

MICROSOFT VISUAL C++ 2012 RUNTIME LIBRARIES

本许可条款是 Microsoft Corporation（或您所在地的 Microsoft Corporation 关联公司）与您之间达成的协议。请阅读本条款的内容。本条款适用于上

☐ 我同意许可条款和条件(A)

Fig 4 Microsoft Visual C++ Redistributable Package

2.4. *C++/CLI*

C++/CLI is a separate programming language that can be viewed as an extension to C++ syntax, but at the same time, it’s one of the programming languages of the .NET platform, just like C# or Visual Basic .NET. It is designed and implemented by Microsoft, so it is not portable to Linux or other systems. It requires Microsoft Visual C++ IDE to compile and debug code, so there is no alternative compiler like GNU Compiler Collection (GCC) available for this language. The good news is that the language is supported in Visual Studio 2005, 2008, and 2010, including corresponding versions of the free Visual C++ Express Edition.

2.5. *Mean Absolute Percentage Error (MAPE)*

MAPE also is known as Mean Absolute Percentage Deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation [9]. It usually expresses the accuracy as a ratio defined by Equation (17):

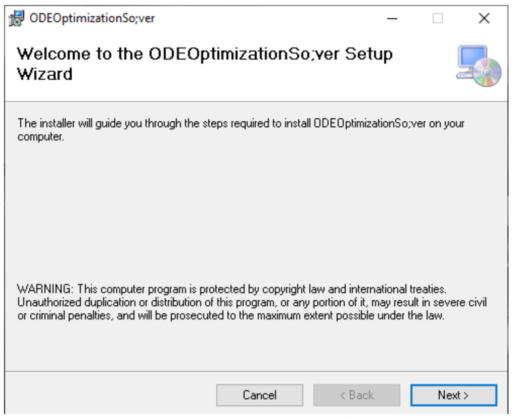
$$\text{MAPE} == \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \qquad (17)$$

where A_t is the actual value and F_t is the forecasted value.
The MAPE is also sometimes reported as a percentage by multiplying it by 100. The difference between A_t and F_t is divided by the actual value, A_t again. The absolute value in this calculation is summed for every forecasted value and divided by the number of fitted points, n. Multiplying by 100% makes it a percentage error [10].

III. MAIN RESULTS

3.1. *Introduction*

This section presents the developed software and how to use it. The application has two tabs, one for Ordinary Differential Equations (ODEs) and the other for Optimization techniques.
To use the software, you have to;first, install it on your computer as shown in Figure 5.


ODEOptimizationSo;ver

Welcome to the ODEOptimizationSo;ver Setup Wizard

The installer will guide you through the steps required to install ODEOptimizationSo;ver on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Fig 5 Installing the Software

3.2. Launching the Software

When the application is launched (opened) by locating the icon on your desktop and clicking on it, you will be welcomed by the welcome screen (loader), after which the main window containing the two tabs will open. ODE is the active tab when the main window opens. This reveals the four methods under ODEs implemented in this project. To use the optimization techniques window, simply click on the tab named Numerical Optimization, located at the top-left corner of the main window, and all the methods will reveal.

To use any of the methods, you just have to locate its tab and on the left pane, click on the method's name and it will open within that same window.

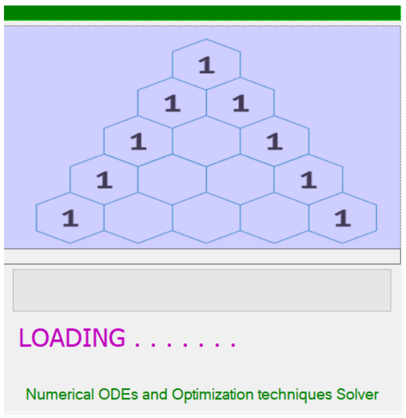


Fig 6 Welcome Screen

Fig 7 the Software's Icon

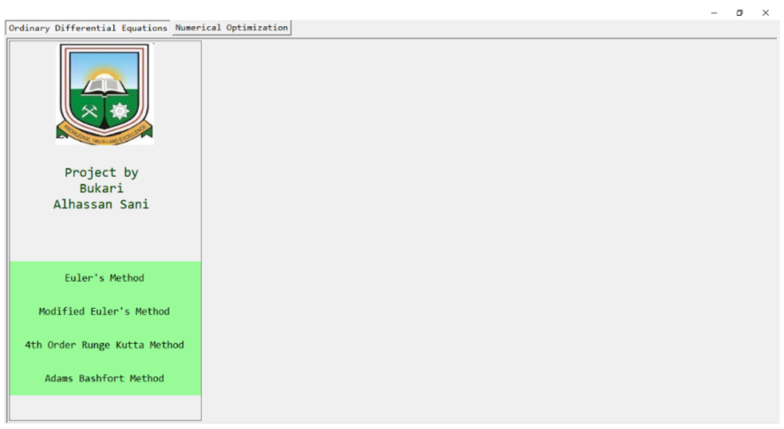


Fig 8 ODEs Tab

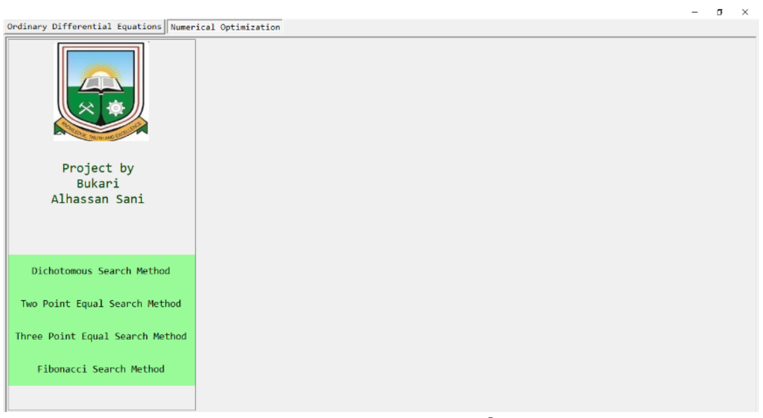

Ordinary Differential Equations
Numerical Optimization
Project by
Bukari
Alhassan Sani
Dichotomous Search Method
Two Point Equal Search Method
Three Point Equal Search Method
Fibonacci Search Method

Fig 9 Optimization Tab

3.3. *Ordinary Differential Equations tab*

This tab contains the Euler's Method, Modified Euler's method, Fourth-order Runge-Kutta's method, and Adams-Bashforth's method. To use any of these methods, you simply have to click on its name and it will open and ready to accept inputs.

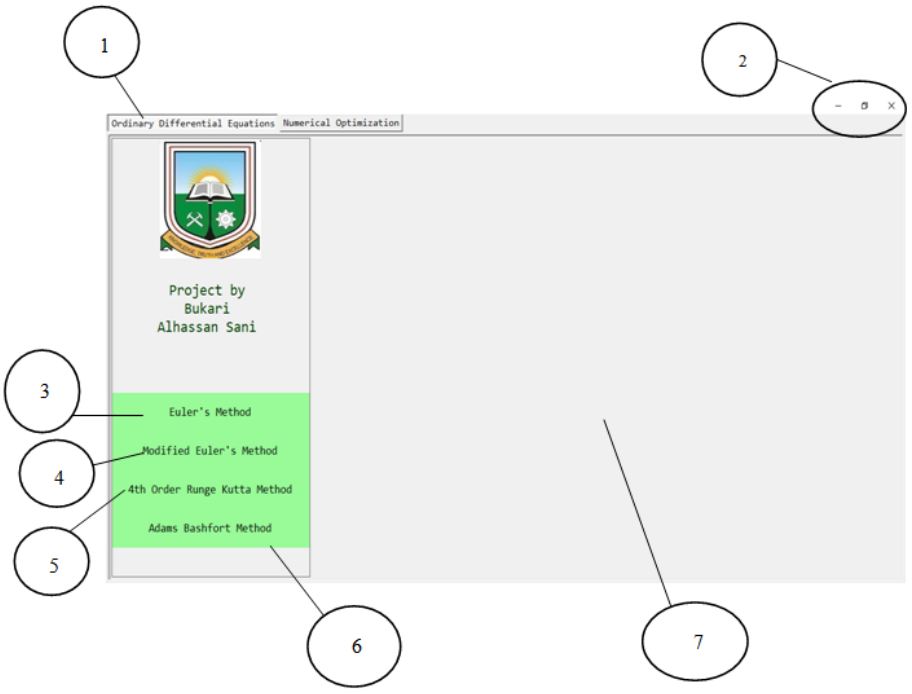


Fig 10 Definition of Labels for the ODEs Tab

*Table 1*Definition of labels for ODEs tab

Number	Name	Description
1	Tab heading	Indicates the current tab you are working on
2	Minimize, maximize and close box	Minimizes maximizes and closes the application
3	Euler's method button	Opens the Euler's method in the panel labeled 7
4	Modified Euler's method button	Opens the Modified Euler's method in the panel labeled 7
5	Fourth-order Runge-Kutta method button	Opens the Fourth order Runge-Kutta method in the panel labeled 7
6	Adams-Bashfort method button	Opens the Adams-Bashfort method in the panel labeled 7
7	Panel	Contains the selected method

3.2.1 *Euler's Method*

Figure 4.7 shows the interface of Euler's Method. It takes an equation of the form *f(x,y)*, x_0, y_0, h and x_n, from the user, where *f(x,y)* is the differential equation, x_0 (lower range) and y_0 are initial conditions for the differential equation, h is the step size and x_n is the upper range. The interface for the Euler's method is the same as that for the Modified Euler and fourth-order Runge-Kutta's method.

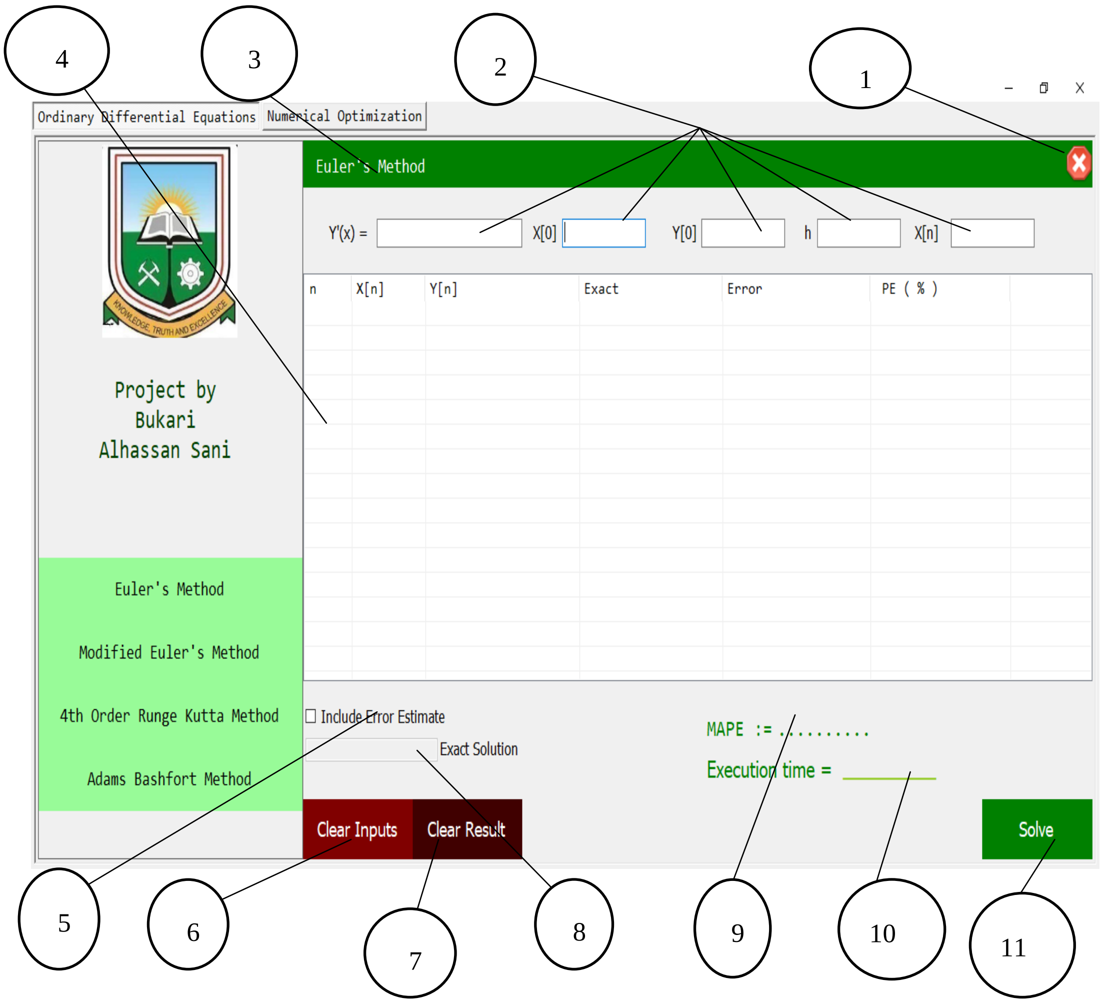
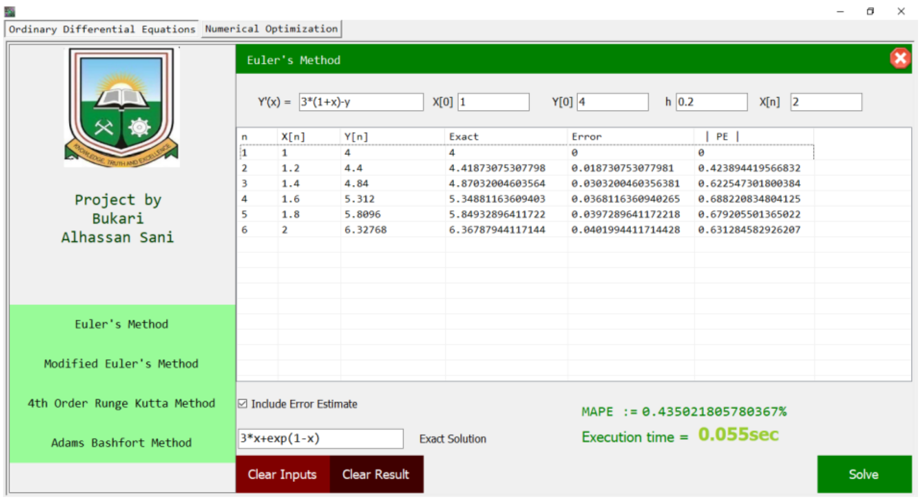


Fig 11 Definition of Labels for Euler's Method

Table 2 Definition of Controls and Labels for the Euler's Method

Number	Name	Description
1	Close button	Closes the current method you are working on.
2	Text boxes	This is where the user inputs the equation, x_0, y_0, h and x_n respectively.
3	Label	Displays the current method's name.

4	List view	This is where the solution is displayed.
5	Checkbox	It toggles the exact equations' textbox (8) active and inactive.
6	Clear inputs button	Clears the content of the textboxes.
7	Button clear result	Clears the list view (4).
8	Textbox	This is where you input the exact equation for the differential equation.
9	Label	This is where the Mean Absolute Percentage Error (MAPE) is displayed in percentage.
10	Label	Displays the time taken (in seconds) to solve the given equation.
11	Button solve	Solves the differential equation and displays the result in the list view (4).



Y'(x) = 3*(1+x)-y X[0] 1 Y[0] 4 h 0.2 X[n] 2

n	X[n]	Y[n]	Exact	Error	| PE |
1	1	4	4	0	0
2	1.2	4.4	4.41873075307798	0.018730753077981	0.423894419566832
3	1.4	4.84	4.87032004603564	0.0303200460356381	0.622547301800384
4	1.6	5.312	5.34881163609403	0.0368116360940265	0.688220834804125
5	1.8	5.8096	5.84932896411722	0.0397289641172218	0.679205501365022
6	2	6.32768	6.36787944117144	0.0401994411714428	0.631284582926207

☑ Include Error Estimate

3*x+exp(1-x) Exact Solution

MAPE := 0.435021805780367%

Execution time = 0.055sec

Fig 12 Solved Example with Euler'sMethod

3.3.1. *Modified Euler's Method*

The interface of the modified Euler's method is similar to that of Euler's method. Figure 16 and 17 shows the interface for the modified Euler's method and example respectively.

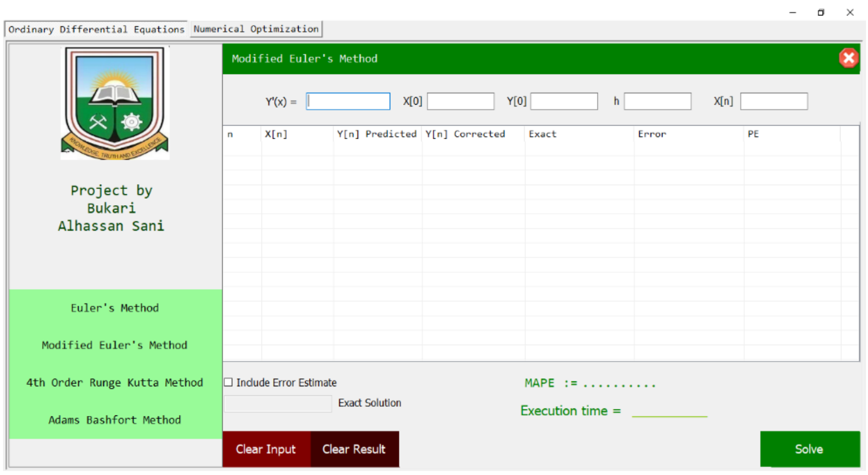
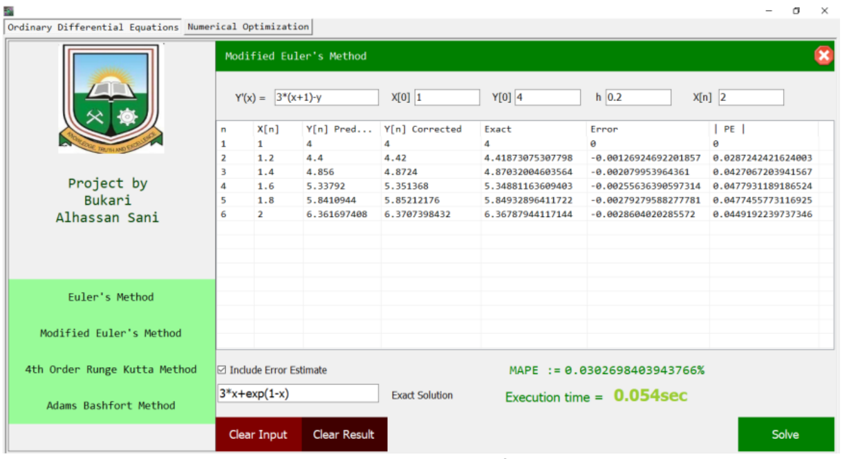


Fig 13 Interface of the Modified Euler's Method



Modified Euler's Method

Y'(x) = 3*(x+1)-y X[0] 1 Y[0] 4 h 0.2 X[n] 2

n	X[n]	Y[n] Pred...	Y[n] Corrected	Exact	Error	| PE |
1	1	4	4	4	0	0
2	1.2	4.4	4.42	4.41873075307798	-0.00126924692201857	0.0287242421624003
3	1.4	4.856	4.8724	4.87032004603564	-0.002079953964361	0.0427067203941567
4	1.6	5.33792	5.351368	5.34881163609403	-0.00255636390597314	0.0477931189186524
5	1.8	5.8410944	5.85212176	5.84932896411722	-0.00279279588277781	0.0477455773116925
6	2	6.361697408	6.3707398432	6.36787944117144	-0.0028604020285572	0.0449192239737346

☑ Include Error Estimate

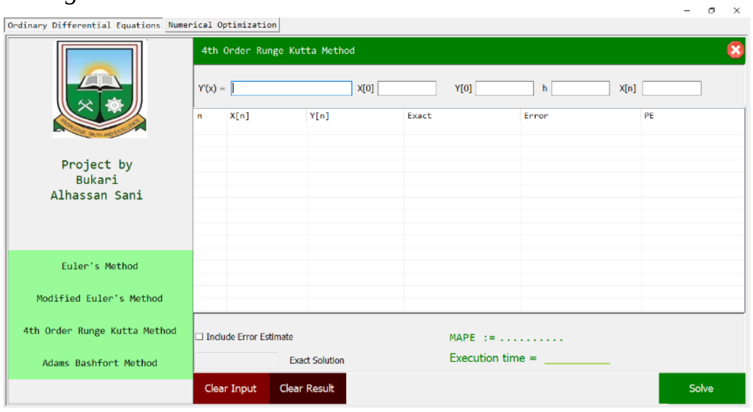
3*x+exp(1-x) Exact Solution

MAPE := 0.0302698403943766%

Execution time = 0.054sec

Fig 14 Solved Example with Modified Euler's Method

3.3.2. *Fourth Order Runge-Kutta Method*



4th Order Runge Kutta Method

Y'(x) = X[0] Y[0] h X[n]

n	X[n]	Y[n]	Exact	Error	PE

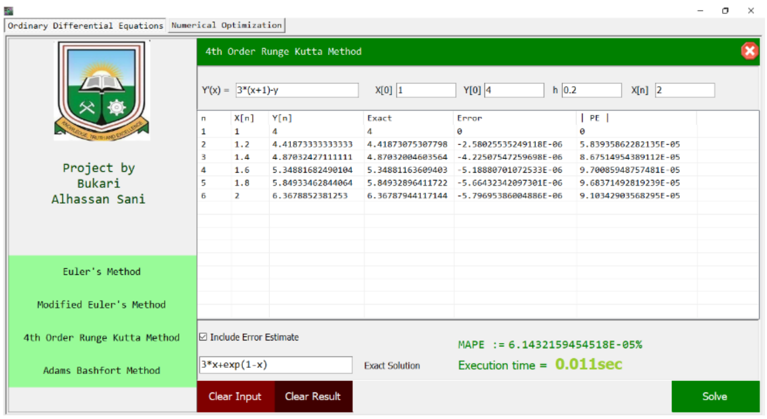
☐ Include Error Estimate

Exact Solution

MAPE := ....

Execution time = ________

Fig 15 Interface of the Runge-Kutta method



4th Order Runge Kutta Method

Y'(x) = 3*(x+1)-y X[0] 1 Y[0] 4 h 0.2 X[n] 2

n	X[n]	Y[n]	Exact	Error	| PE |
1	1	4	4	0	0
2	1.2	4.41873333333333	4.41873075307798	-2.58025535249118E-06	5.83935862282135E-05
3	1.4	4.87032427111111	4.87032004603564	-4.22507547259698E-06	8.67514954389112E-05
4	1.6	5.34881682490104	5.34881163609403	-5.18880701072533E-06	9.70085948757481E-05
5	1.8	5.84933462844064	5.84932896411722	-5.66432342097301E-06	9.68371492819239E-05
6	2	6.3678852381253	6.36787944117144	-5.79695386004886E-06	9.10342903568295E-05

☑ Include Error Estimate

3*x+exp(1-x) Exact Solution

MAPE := 6.1432159454518E-05%

Execution time = 0.011sec

Fig 16 Solved Example with the Runge-Kutta Method

3.3.3. *Adams-Basford Predictor-Corrector Method*

The interface for this method is almost the same as the Euler, Modified Euler, and Runge-Kutta method. The interface has additional labels; y1, y2, and y3, which are the first-three y-values from the Runge-Kutta method.

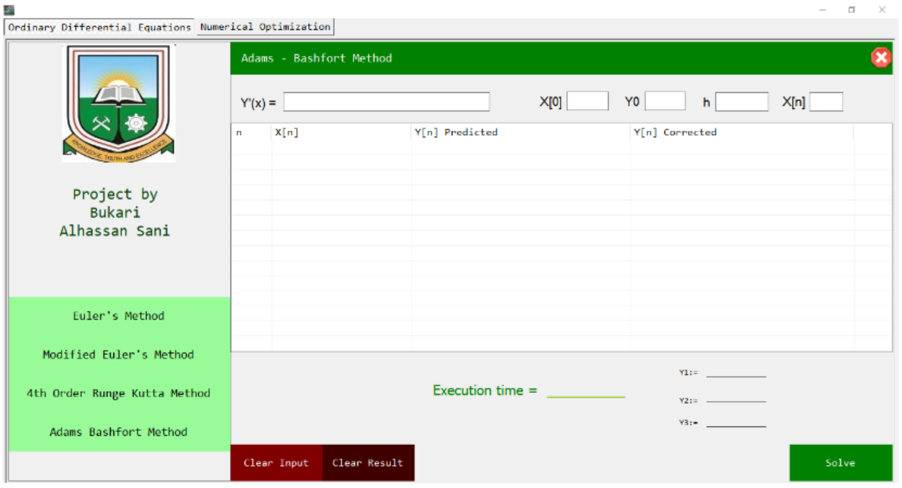
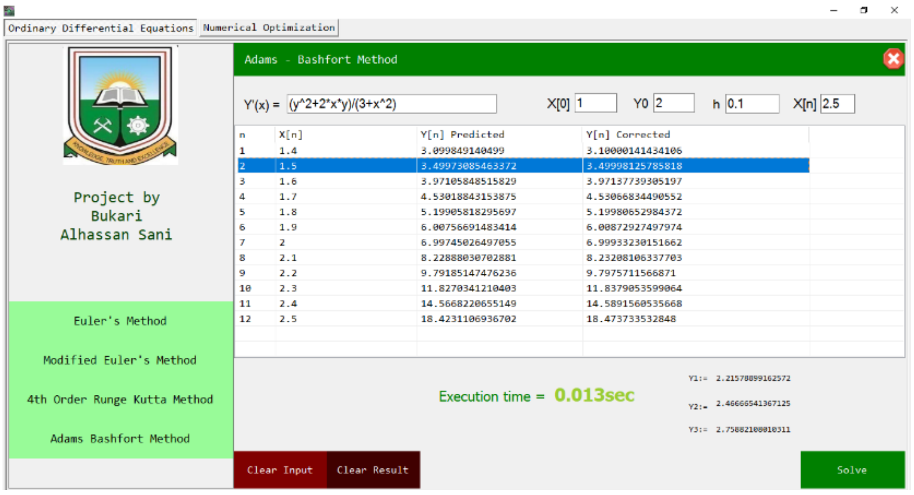


Fig 17 Interface of the Adams Method



Y'(x) = (y^2+2*x*y)/(3+x^2)   X[0] 1   Y0 2   h 0.1   X[n] 2.5

n	X[n]	Y[n] Predicted	Y[n] Corrected
1	1.4	3.099849140499	3.10000141434106
2	1.5	3.49973085463372	3.49998125785818
3	1.6	3.97105848515829	3.97137739305197
4	1.7	4.53018843153875	4.53066834490552
5	1.8	5.19905818295697	5.19980652984372
6	1.9	6.00756691483414	6.00872927497974
7	2	6.99745026497055	6.99933230151662
8	2.1	8.22888030702881	8.23208106337703
9	2.2	9.79185147476236	9.7975711566871
10	2.3	11.8270341210403	11.8379053599064
11	2.4	14.5668220655149	14.5891560535668
12	2.5	18.4231106936702	18.473733532848

Execution time = 0.013sec

Y1:= 2.21578899162572
Y2:= 2.46666541367125
Y3:= 2.75882100010311

Fig 18 Solved Example with Adam’s Method

3.3 *Numerical Optimization’s Tab*

This tab contains on the left side, the Dichotomous search method, two-point equal search method, three-point equal search method, and the Fibonacci search method.

3.3.1 *Dichotomous Search Method*

This tab has the same number of methods (4) as that of the ODEs tab. To use any of the methods, click on it and it will open in the left panel.

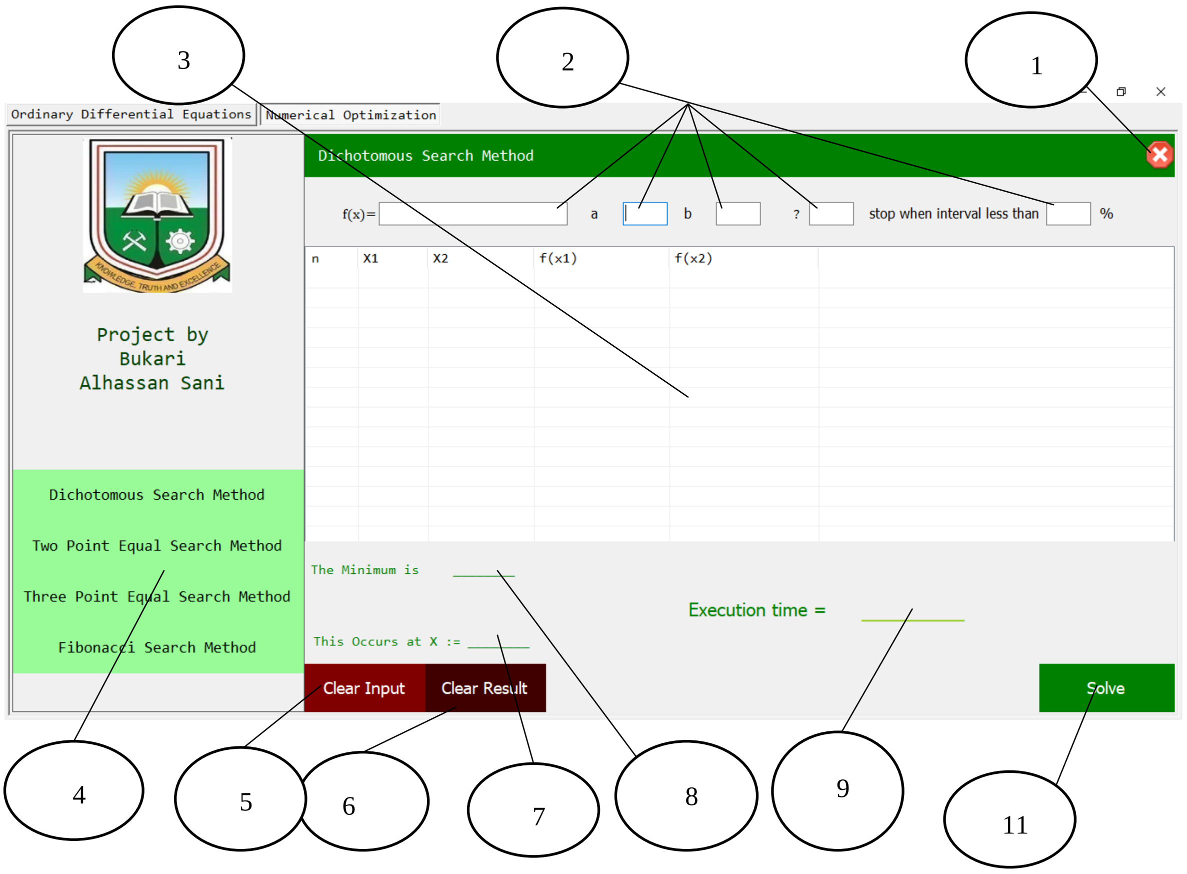
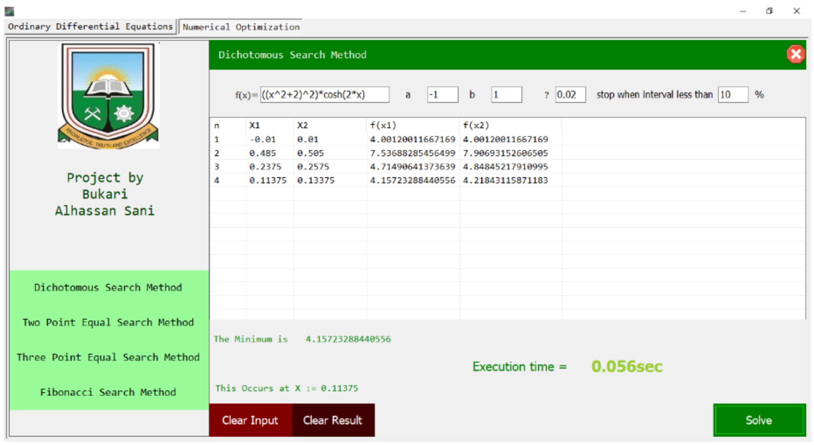


Figure 19 Interface of Dichotomous Search Method

Table 3Definition of Labels for the Dichotomous Search Method

Number	Name	Description
1	Close button	Closes the current method opened.
2	Textboxes	This is where the user inputs the function, a, b, epsilon, and the percentage the initial interval is to be minimized too.
3	List view	Displays the result
4	Buttons	This is where the user changes the method of interest.
5	Clear input button	Clears the textboxes and gives focus to the first textbox.
6	Clear result button	Clears the content of the list view.
7	Label	Displays the value of x where the minimum functional value occurs.
8	Label	Displays the minimum functional value
9	Label	Shows the time taken in seconds to solve and display the results in the list view.
10	Solve button	Solves the given function.


Ordinary Differential Equations
Numerical Optimization
Dichotomous Search Method
Project by
Bukari
Alhassan Sani
Dichotomous Search Method
Two Point Equal Search Method
Three Point Equal Search Method
Fibonacci Search Method
Clear Input
Clear Result
Solve

f(x)= ((x^2+2)^2)*cosh(2*x)   a -1   b 1   ? 0.02   stop when interval less than 10 %

```
n   X1        X2        f(x1)             f(x2)
1   -0.01     0.01      4.00120011667169  4.00120011667169
2   0.485     0.505     7.53688285456499  7.90693152606505
3   0.2375    0.2575    4.71490641373639  4.84845217910995
4   0.11375   0.13375   4.15723288440556  4.21843115871183
```

The Minimum is 4.15723288440556

Execution time = 0.056sec

This Occurs at X := 0.11375

Fig 20 Solved Example with the Dichotomous Search Method

3.3.2 Two-Point Equal Search Method

This has the same controls and labels as that of the Dichotomous method, except for the fact that it does not make use of epsilon in the computation, hence that textbox is missing.

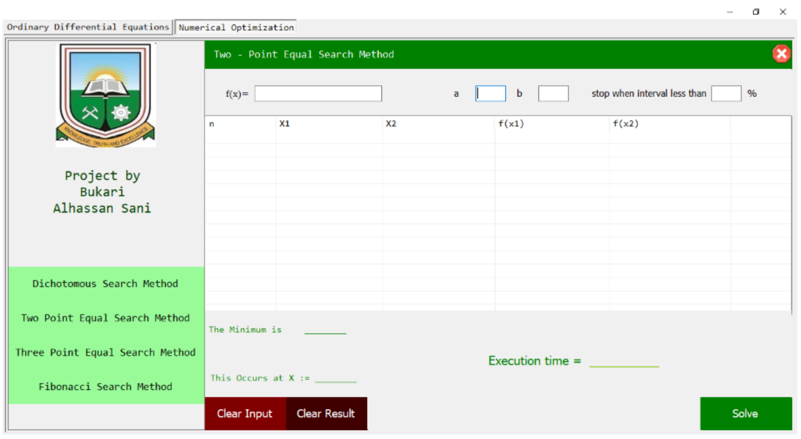
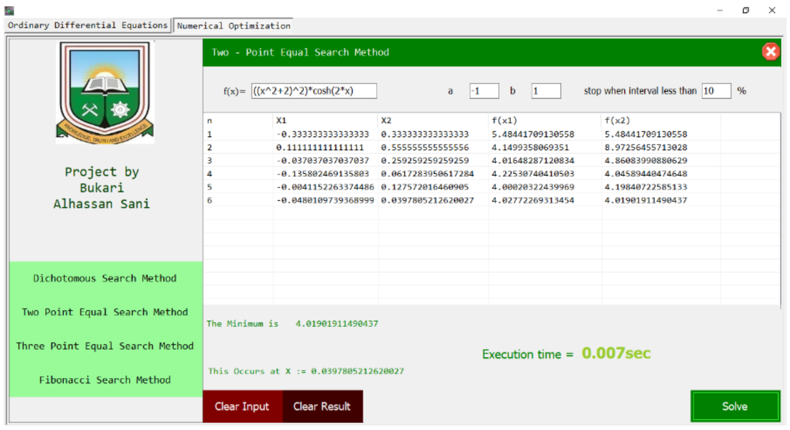

Ordinary Differential Equations
Numerical Optimization
Project by Bukari Alhassan Sani
Dichotomous Search Method
Two Point Equal Search Method
Three Point Equal Search Method
Fibonacci Search Method
Two - Point Equal Search Method
f(x)=
a
b
stop when interval less than
%
n
X1
X2
f(x1)
f(x2)
The Minimum is
This Occurs at X :=
Execution time =
Clear Input
Clear Result
Solve

Fig 21 Interface Two-Point Equal Search Method


Ordinary Differential Equations
Numerical Optimization
Project by Bukari Alhassan Sani
Dichotomous Search Method
Two Point Equal Search Method
Three Point Equal Search Method
Fibonacci Search Method
Two - Point Equal Search Method
Clear Input
Clear Result
Solve

f(x)= ((x^2+2)^2)*cosh(2*x) a -1 b 1 stop when interval less than 10 %

n	X1	X2	f(x1)	f(x2)
1	-0.333333333333333	0.333333333333333	5.48441709130558	5.48441709130558
2	0.111111111111111	0.555555555555556	4.1499358069351	8.97256455713028
3	-0.037037037037037	0.259259259259259	4.01648287120834	4.86083990880629
4	-0.135802469135803	0.0617283950617284	4.22530740410503	4.04589440474648
5	-0.0041152263374486	0.127572016460905	4.00020322439969	4.19840722585133
6	-0.0480109739368999	0.0397805212620027	4.02772269313454	4.01901911490437

The Minimum is 4.01901911490437

This Occurs at X := 0.0397805212620027

Execution time = 0.007sec

Fig 22 Solved Example with the Two-Point Equal Search Method

3.3.3 Three-Point Equal Search Method

The interface for the three-point method and the two-point method use the same controls and labels.

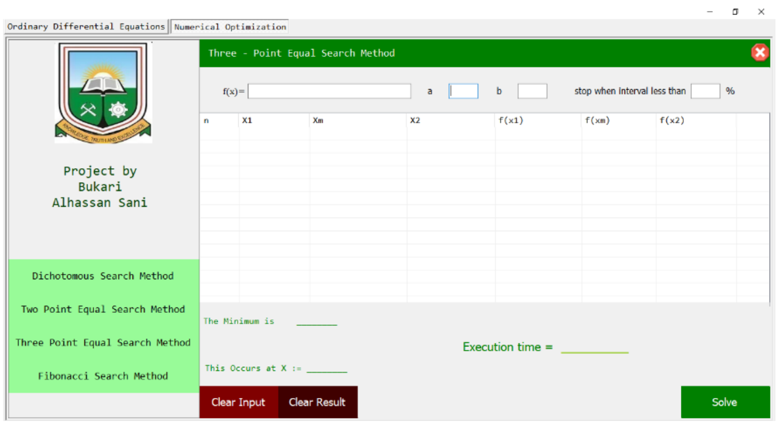
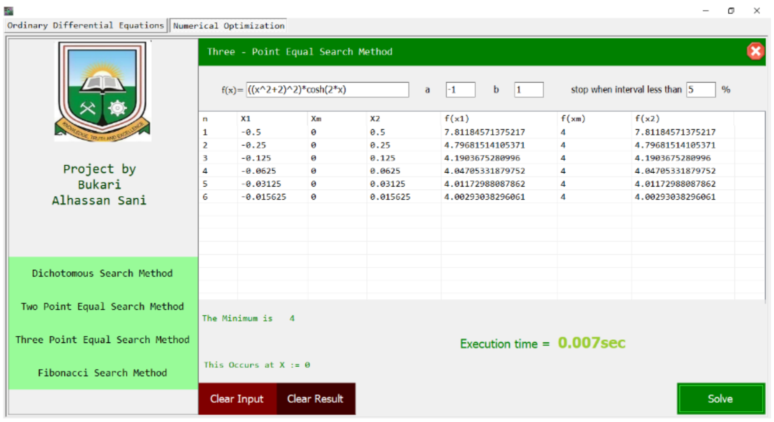

Ordinary Differential Equations
Numerical Optimization
Project by Bukari Alhassan Sani
Dichotomous Search Method
Two Point Equal Search Method
Three Point Equal Search Method
Fibonacci Search Method
Three - Point Equal Search Method
f(x)=
a
b
stop when interval less than
%
n
X1
Xm
X2
f(x1)
f(xm)
f(x2)
The Minimum is
This Occurs at X :=
Execution time =
Clear Input
Clear Result
Solve

Fig 23 Interface of Three-Point Equal Search Method


Ordinary Differential Equations
Numerical Optimization
Project by Bukari Alhassan Sani
Dichotomous Search Method
Two Point Equal Search Method
Three Point Equal Search Method
Fibonacci Search Method
Clear Input
Clear Result
Solve

Three - Point Equal Search Method

f(x)= ((x^2+2)^2)*cosh(2*x)   a -1   b 1   stop when interval less than 5 %

n	X1	Xm	X2	f(x1)	f(xm)	f(x2)
1	-0.5	0	0.5	7.81184571375217	4	7.81184571375217
2	-0.25	0	0.25	4.79681514105371	4	4.79681514105371
3	-0.125	0	0.125	4.1903675280996	4	4.1903675280996
4	-0.0625	0	0.0625	4.04705331879752	4	4.04705331879752
5	-0.03125	0	0.03125	4.01172988087862	4	4.01172988087862
6	-0.015625	0	0.015625	4.00293038296061	4	4.00293038296061

The Minimum is 4

This Occurs at X := 0

Execution time = 0.007sec

Fig 24 Solved Example with the Three-Point Equal Search Method

3.3.4 Fibonacci Search Method

The interface for this method is a bit different in that, in addition to the controls, it has a button that triggers a form that helps the user to determine the number of applications b$_y$ generating Fibonacci numbers. Given r% as the percentage the initial interval is to be reduced to, we calculate $F_n > \frac{1}{r\%}$ and choose the corresponding n as the number of applications.

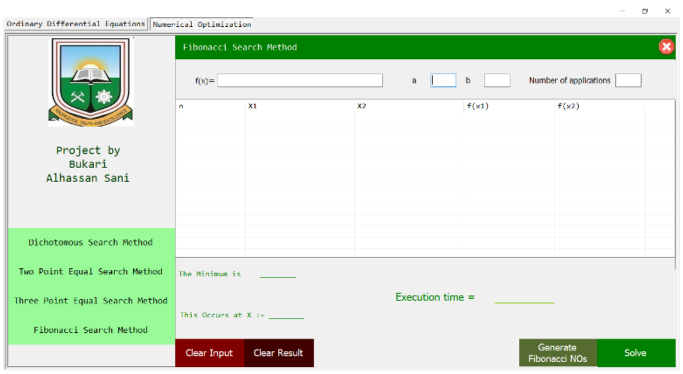
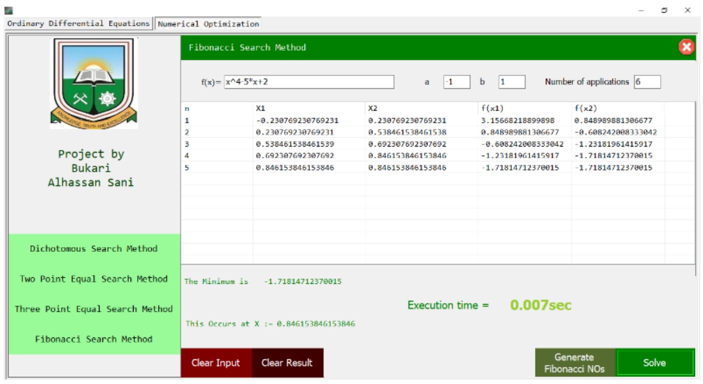


Fig 25 Interface of Fibonacci Search Method


Ordinary Differential Equations
Numerical Optimization
Project by Bukari Alhassan Sani
Dichotomous Search Method
Two Point Equal Search Method
Three Point Equal Search Method
Fibonacci Search Method
Clear Input
Clear Result
Generate Fibonacci NOs
Solve

Fibonacci Search Method

f(x)= x^4-5*x+2   a -1   b 1   Number of applications 6

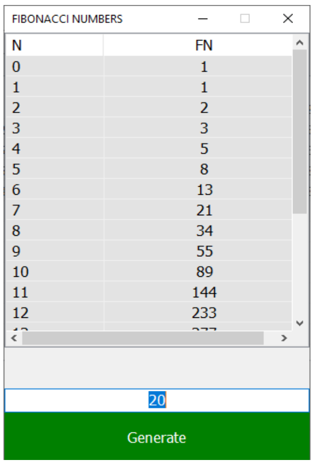
n	X1	X2	f(x1)	f(x2)
1	-0.230769230769231	0.230769230769231	3.15668218899898	0.848989881306677
2	0.230769230769231	0.538461538461538	0.848989881306677	-0.608242008333042
3	0.538461538461539	0.692307692307692	-0.608242008333042	-1.23181961415917
4	0.692307692307692	0.846153846153846	-1.23181961415917	-1.71814712370015
5	0.846153846153846	0.846153846153846	-1.71814712370015	-1.71814712370015

The Minimum is -1.71814712370015

This Occurs at X := 0.846153846153846

Execution time = 0.007sec

Fig 26 Solved Example with the Fibonacci Search Method



N	FN
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	89
11	144
12	233

Fig 27 Fibonacci number Generator

IV CONCLUSION

In conclusion, a desktop application for some selected methods in Ordinary Differential Equations and sequential search methods in Optimization techniques have been developed. The time is taken by each of the methods used to execute (solve) given functions has also been shown. The error associated with methods like Euler, Modified Euler, and the fourth-order Runge-Kutta can also be determined if the user supplies the application with the exact solution of the differential equation. Mean Absolute Percentage Error (MAPE) as a measure of predicting accuracy of a forecasting method was used to determine how much (in percentage) these methods vary from the exact solution.

Knowledge in programming is not required to be able to use the application since the interface is quite simple and self-explanatory.

REFERENCES

[1]. Jeffery, A. (2002), "Advanced Engineering Mathematics", pp. 1123.
[2]. Smith, J.D. (2004), "Three Applications of Optimization in Computer Graphics".
[3]. Bondarev, A.E.E., Galaktionov, V.A., and Chechetkin, V.M., (2011),"Analysis of the Development Concepts and Methods of Visual Datare Presentation in Computational Physics", *Computational Mathematics And Mathematical Physics*, *51*(4), pp.624-636.
[4]. Salleh, S., Zomaya, A.Y. and Bakar, S.A. (2007), "Computing For Numerical Methods Using Visual C++".
[5]. Butcher, J.C. and Goodwin, N., (2008), "Numerical Methods for Ordinary Differential Equations*",* Vol. 2. New York: Wiley.
[6]. Hussain, K., Ismail, F. and Senu, N. (2016), "Solving Directly Special Fourth-Order Ordinary Differential Equations Using Runge–Kutta Type Method". *Journal of Computational and Applied Mathematics*, *306*, pp.179-199.
[7]. Pardalos, P. (Eds.) (2008). "Encyclopedia of Optimization. Boston: Springer". *International Journal of Business Information Systems* pp. 1538–1542
[8]. Johnson, B., Young, M. and Skibo, C., (2002),"Inside Microsoft Visual Studio". *NET*. Microsoft Press.
[9]. Hyndman, Rob J., and Anne B. (2006), "Another Look at Measures of Forecast Accuracy". *International Journal of Forecasting*, Vol 22, No.4, pp. 679-688.
[10]. Khair, U., Fahmi, H., Al Hakim, S. and Rahim, R. (2017), "Forecasting Error Calculation with Mean Absolute Deviation and Mean Absolute Percentage Error". *In Journal Of Physics: Conference Series*, Vol. 930, No. 1, IOP Publishing.

AUTHOR'S PROFILE

First Author

BoahAhobaMasha (2020) is an assistant lecturerand a PHD candidate in the Department of Mathematical Sciences, University of Mines and Technology, Ghana. Email: mabuah@umat.edu.gh

Second Author

Alhassan Sani Bukari (2020) is a student in the Department of Mathematical Sciences, University of Mines and Technology, Ghana. Email: bukarisani@gmail.com

Third Author

BoaheneBoahemaa Martha (2020) is a student in the Department of Mathematical Sciences, University of Mines and Technology, Ghana. Email: boahenemartha7@gmail.com