

**Green University of Bangladesh**

**Department of Electrical and Electronic Engineering**

**EEE- 302**

**Numerical Technique Laboratory**

|  |  |
| --- | --- |
| Student ID |  |
| Student Name |  |
| Section |  |
| Name of the Program |  |
| Name of the Department |  |

|  |  |  |
| --- | --- | --- |
| **CONTENTS** | | Page No |
|  | Instructions for Laboratory | 3 |
|  | Laboratory Course Syllabus | 4 |
| Experiment -1 | Introduction to MATLAB (Part-A) | 9 |
| Experiment- 2 | Introduction to MATLAB (Part-B) | 13 |
| Experiment- 3 | Iterative Processes For Root Finding  (Iterative Method, Aitken’s ∆² Acceleration Method, Bisection Method) | 19 |
| Experiment- 4 | Solutions to Non-Linear Equations  (Secant Method & Regula Falsi Method) | 25 |
| Experiment -5 | Interpolation by Newton-Gregory forward difference formula | 29 |
| Experiment- 6 | Interpolation by Newton-Gregory backward difference formula | 33 |
| Experiment- 7 | Numerical Differentiation  (Based on Newton-Gregory Forward & Backward Differences) | 37 |
| Experiment -8 | Numerical Differentiation (Based on Lagrange Interpolation) &  Numerical Integration (Based on Simple Trapezium Rule) | 43 |
| Experiment -9 | Numerical Integration By Simpson's 1/3, 3/8th Rule & Romberg Integration | 49 |
| Experiment- 10 | Solution of system of Linear Equations By  A direct method (LU Decomposition),  An indirect method (Jacobi or Gauss Seidel Method) | 55 |

**INSTRUCTIONS FOR LABORATORY**

* The experiments are designed to illustrate about different arenas control system, communication, solid-state design, robotics, mechatronics, Aeronautics etc. Conduct the experiments with interest and an attitude of learning.
* Students should come with thorough preparation for the experiment to be conducted.
* Students should come with proper dress code.
* Students will not be permitted to attend the laboratory unless they bring the practical record fully completed in all respects pertaining to the experiment conducted in the previous class.
* Work quietly and carefully (the whole purpose of experimentation is to develop logic for MATLAB programming!) and equally share the work with your partners.
* Be honest in developing and representing your program. If a particular MATLAB output appears wrong repeat the program carefully.
* All presentations of programs, outputs and graphs calculations should be neatly and carefully done.
* Graphs should be carefully drawn by MATLAB program. Always label graphs and the axes and display units in the program.
* If you finish early, spend the remaining time to complete the laboratory report writing. Come equipped with calculator, scales, pencils etc.
* Handle instruments with care. Report any breakage or faulty equipment to the Instructor. Shutdown your computer you have used for the purpose of your experiment before leaving the Laboratory.

**GREEN UNIVERSITY OF BANGLADESH (GUB)**

**COURSE SYLLABUS**

|  |  |  |
| --- | --- | --- |
| 1 | **Faculty** | Faculty of Science & Engineering |
| 2 | **Department** | Department of EEE |
| 3 | **Programme** | BSEEE [BSc in Electrical & Electronic Engineering] |
| 4 | **Name of Course** | Numerical Technique Laboratory |
| 5 | **Course Code** | EEE 404 |
| 6 | **Trimester and**  **Year** |  |
| 7 | **Pre-requisites** | EEE 301, EEE 406 & Math 101 |
| 8 | **Status** | Core EEE Course |
| 9 | **Credit Hours** | 1.5 |
| 10 | **Section** |  |
| 11 | **Class Hours** |  |
| 12 | **Class Location** | Room: 908 (Simulation Lab.) |
| 13 | **Course website** |  |
| 14 | **Name (s) of Academic staff / Instructor(s)** |  |
| 15 | **Contact** |  |
| 16 | **Office** |  |
| 17 | **Counselling Hours** |  |
| 18 | **Text Book** | Steven C. Chapra & Raymond P. Canale, Numerical Methods for Engineers, Sixth Edition |
| 19 | **Reference** | •Applied Numerical Methods with MATLAB (for Engineers & Scientists)  (Special Indian Edition),  Steven C. Chapra  •Numerical Methods,  S. Balachandra Rao, C.K.Shantha |
| 20 | **Equipment & Aids** | Bring your notebook and calculator. Equipment will be provided in the laboratory and MATLAB R2017a software is installed in the respective laboratory computers. Do collect the software named ‘MATLAB R2017a´ for home practice. |
| 21 | **Course Rationale** | EEE 404 is one of the fundamental courses for EEE students. It aims to give students the practical idea about statistical problem solving such as *pdf*, *cdf* and how to analyse them. It is a pre-requisite for many other courses in EEE. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 22 | **Course Description** | Numerical Technique Lab. is an attractive course for the arenas control system, communication, solid-state design, robotics, mechatronics, Aeronautics etc. Its applications are quite widely seen. This course will discuss in advance level about the fundamental and primitive ideas of control system, signals, systems and a brief application about them in Digital domain. | | | |
| 23 | **Course Objectives** | The course is designed to provide the background of the following topics  1. Explain basic features configurations and applications of some signals in digital domain.  2. Familiar with the practical implementation of digital signals.  3. Explain different digital systems and their properties.  4. Explain a system through difference equation in digital domain.  5. Manipulate frequency response of a system in digital domain.  6. Justify the system transfer function through different transformation techniques in digital domain.  7. Recognize the applications of Time and Frequency domain analysis and advanced signal processing aspects | | | |
| 24 | **Learning Outcomes** | After the end of this course, the students will be able to:  1. Explain different signals and systems in digital domain.  2. Explain features, configurations of various systems in digital domain.  3. Utilize system properties through Time-domain analysis and Frequency-domain analysis in digital domain.  4. Measure system transfer function from the different transformations analysis in digital domain.  5. Explain the applications of different transformations analysis in digital domain.  6. Recognize these concepts on courses like Advanced Numerical Methods, Control Systems, communication theory etc. | | | |
| 25 | **Teaching Methods** | Lecture, Laboratory hardware and software experiments, Project Developments. | | | |
| 26 | **Topic Outline** |  | | | |
|  | **Class** | **Topics Or**  **Assignments** | **CLOs** | **Reading Reference** | **Activities** |
|  | 1 | Introduction to MATLAB (Part-A) | 1,2,4 |  | Laboratory Experiment |
|  | 2 | Introduction to MATLAB (Part-B) | 3 |  | Laboratory Experiment |
|  | 3 | Iterative Processes For Root Finding (Iterative Method, Aitken’s ∆² Acceleration Method, Bisection Method) | 1,2,3,4 |  | Laboratory Experiment |
|  | 4 | Solutions to Non-Linear Equations (Secant Method & Regula Falsi Method) | 1,2,3,4 |  | Laboratory Experiment |
|  | 5 | Interpolation by Newton-Gregory forward difference formula | 1,2,3,4 |  | Laboratory Experiment |
|  | 6 | Interpolation by Newton-Gregory backward difference formula | 1,2,3,4 |  | Laboratory Experiment |
|  | 7 | MID TERM EXAMINATION |  |  | Quiz, Lab test, Viva |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 8 | Numerical Differentiation  (Based on Newton-Gregory Forward & Backward Differences) | 1,2,3,4 |  | Laboratory Experiment |
|  | 9 | Numerical Differentiation (Based on Lagrange Interpolation) &  Numerical Integration (Based on Simple Trapezium Rule) | 1,2,3,4 |  | Laboratory Experiment |
|  | 10 | Numerical Integration By Simpson's 1/3, 3/8th Rule & Romberg Integration | 1,2,3,4 |  | Laboratory Experiment |
|  | 11 | Solution of system of Linear Equations By a direct method (LU Decomposition) & an indirect method (Jacobi or Gauss Seidel Method) | 1,2,3,4 |  | Laboratory Experiment |
|  | 12 | FINAL TERM EXAMINATION (Written Lab Exam, Viva, Performance Test) |  |  | Quiz, Lab test, Viva |
|  | 13 | Project presentation | 1,2,3,4 |  | Presentation and Report Submission |
| 27 | **Assessment Methods** | |  |  | | --- | --- | | **Assessment Types** | **Marks** | | Attendance and Participation (Class Room & Course Page) | 10% | | Viva | 15% | | Performance Test | 25% | | Lab. Report | 20% | | Final Exam | 30% | | **Total** | **100%** | | | | |
| 28 | **Grading Policy** | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Letter Grade** | **Marks %** | **Grade Point** | **Letter Grade** | **Marks%** | **Grade Point** | | A+ (Plus) | 80-100 | 4.00 | C+ (Plus) | 50-54 | 2.50 | | A (Plain) | 75-79 | 3.75 | C (Plain) | 45-49 | 2.25 | | A- (Minus) | 70-74 | 3.50 | D (Plain) | 40-44 | 2.00 | | B+ (Plus) | 65-69 | 3.25 | F (Fail) | <40 | 0.00 | | B (Plain) | 60-64 | 3.00 | I\* | - | Incomplete | | B- (Minus) | 55-59 | 2.75 | W\* | - | Withdrawal | | | | |

|  |  |  |
| --- | --- | --- |
| 29 | **Additional Course** **Policies** | 1. Lab Reports: Report on previous Experiment must be submitted before the beginning of new experiment. A bonus may be obtained if a student submits a neat, clean and complete lab report.  2. Examination: There will be a mid-term exam and final exam both of which will be closed book.  3. Unfair means policy: In case of copying/plagiarism in any of the assessments, the students involved will receive zero marks. Zero Tolerance will be shown in this regard. In case of severe offences, actions will be taken as per university rule.  4. Counseling: Students are expected to follow the counseling hours posted. In case of emergency/unavoidable situations, students can e-mail me to make an appointment. Students are regularly advised to check the eLMS course page for updates/materials.  5. Policy for Absence in Class/Exam: If a student is absent in the class for anything other than medical reasons, he/she will not receive attendance. If a student misses a class for genuine medical reasons, he/she must submit an application with the supporting documents (prescription/medical report). He/she will then have to follow the instructions given by the instructor for make-up. In case of absence in the mid/final exam for medical grounds, the student must also get his/her application forwarded by the head of the department before a make-up exam can be taken.  It is recommended that the students inform the instructor beforehand through mail if they feel that they will miss a class/evaluation due to medical reasons. |
| 30 | **Additional Info.** | 1. Academic Calendar Fall 2017: http://www.green.edu.bd/academics/academic-calendar 2. Academic Information and Policies: http://www.green.edu.bd/academics/academic-rules-a-regulations 3. Grading and Performance Evaluation: http://www.green.edu.bd/academics/academic-rules-a-regulations 4. Proctorial Rules: http://www.green.edu.bd/administrator/proctors-office |

**Experiment No. 1:**

**Name of the Experiment: Introduction to MATLAB (Part A)**

**Objectives:**

1. Explain basic features configurations and applications of some signals in digital domain.
2. Familiar with the practical implementation of digital signals.
3. Explain different digital systems and their properties.
4. Explain a system through difference equation in digital domain.
5. Manipulate frequency response of a system in digital domain.
6. Justify the system transfer function through different transformation techniques in digital domain.
7. Recognize the applications of Time and Frequency domain analysis and advanced signal processing aspects

**Learning Outcome:** After completing this experiment the students will be able to:

1. Explain different signals and systems in digital domain.
2. Explain features, configurations of various systems in digital domain.
3. Utilize system properties through Time-domain analysis and Frequency-domain analysis in digital domain.
4. Measure system transfer function from the different transformations analysis in digital domain.
5. Explain the applications of different transformations analysis in digital domain.
6. Recognize these concepts on courses like Advanced Numerical Methods, Control Systems, communication theory etc.

**Theory:**

Starting MATLAB:

* You can start MATLAB R2017a on Microsoft Windows Platform (Win7/Win10) by double clicking the **MATLAB** shortcut icon on your windows desktop or simply click **MATLAB R2017a** from the start menu.
* As an alternative method, click on the **Start** button then type **matlab** in the ‘search field’ then press **Enter**. MATLAB will start immediately.
* After MATLAB starts, you can change the directory in which MATLAB saves your MATLAB files. To do this, click on **‘…’** button then select the new location.

Desktop Tools:

1. The Command Window: To enter variables, execute commands and to run M-files.
2. Menus:

* File: from the file menu, you can create a new M-file, figure…etc. You can also open any file and you can access the preferences of MATLAB.
* Edit: cut, copy, paste...etc.
* Desktop: to control the desktop of MATLAB.

*Tip*: to restore the default desktop go to **Desktop** →**Desktop Layout**→**default**

* Window: to get access to the windows/files e.g. the open M-files documents.

1. The Current Directory Browser: any files you want to run must either be in the current directory or on search path. The current directory browser enables you to browse all the files saved in the current directory. You can run, rename, delete…etc.
2. Command History: in the command history you can view the previously used functions and copy and execute selected lines.
3. Lunch Pad: provides easy access to tools, demos, and documentations.
4. Help Browser: to search and view documentations for all your **MathWorks** products.
5. Workspace Browser: the MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. To view the workspace and information about each variable, use the Workspace Browser, or use the commands **who** and **whos.**
6. Array Editor: double click on a variable in the Workspace Browser to see it in the Array Editor.
7. Editor/Debugger: to create and debug M-files.

**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |
|  |  |

**Procedure:**

First Steps in MATLAB:

When MATLAB starts, the special >> prompt (the command line) appears, MATLAB is ready to receive your commands. Try to compute c=a+b, where a=10, and b=20.

MATLAB Command:

>> a=10; b=20; c=a+b

Try The Followings:

>> x=[1:10]

>> x=[1:1:10]

>> x=[1:2:10] %[start:step:stop]

>> y=linspace(0,10,5) %linspace(start,stop,no.of data)

>> t=2\*pi\*50 % t=2πf

>> sin(30\*pi/180) % sin(300)

>> asin(1/2)\*180/pi %sin-1(1/2)

>> 10\*exp(-2) % 10e-2

>> 9e-5 %9x10-5

>> date

>> calendar

>> (log10(10))^4 %(log10(10))4

>> (log2(5))^4 %(log2(5))4

>> (log(10))^4 %(loge(10))4

>> x=0:2:16; y=2\*x;

>> t=linspace(0,2\*pi,100);

>> x=sin(t);

>> y=cos(t);

>> plot(x)

>> plot(y)

>> plot(x,y)

>> subplot(3,1,1)

>> plot(x)

>> subplot(3,1,2)

>> plot(y)

>> subplot(3,1,3)

>> plot(x,y)

>> plot(x,'-r')

>> hold on

>> plot(y,'-.b')

>> hold on

>> plot(x,y,':k')

>> legend('Sin','Cos','Circle',0)

**Report Question:**

Solve the followings by MATLAB,

1. sin(2250) + cot(300) + tan-1(1/2) + 10e-10 + 9x10-2 + (log1010)3 + (loge10)5

Ans: 94.6135

1. Think about A and B are the last two digits of your Class Roll:

sin(AB0) + cot(BA0) + tan-1(B/A) + Be-BA + Ax10-A + (log10B)AB + (logeBA)A

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha

**Experiment No. 2:**

**Name of the Experiment: Introduction to MATLAB (Part B)**

**Objectives:**

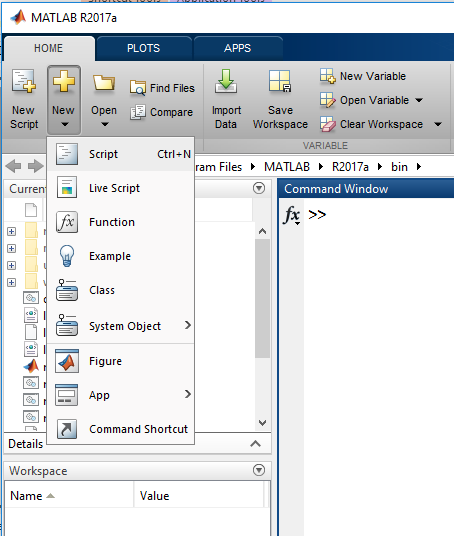
1. Explain basic features configurations and applications of some signals in digital domain.
2. Familiar with the practical implementation of digital signals.
3. Explain different digital systems and their properties.
4. Explain a system through difference equation in digital domain.
5. Manipulate frequency response of a system in digital domain.
6. Justify the system transfer function through different transformation techniques in digital domain.
7. Recognize the applications of Time and Frequency domain analysis and advanced signal processing aspects

**Learning Outcome:** After completing this experiment the students will be able to:

1. Explain different signals and systems in digital domain.
2. Explain features, configurations of various systems in digital domain.
3. Utilize system properties through Time-domain analysis and Frequency-domain analysis in digital domain.
4. Measure system transfer function from the different transformations analysis in digital domain.
5. Explain the applications of different transformations analysis in digital domain.
6. Recognize these concepts on courses like Advanced Numerical Methods, Control Systems, communication theory etc.

**Theory:**

File:

M-files provide an easy way to write and excite your commands and programs. For a large number of commands and complex problem-solving M-files is a must. It allows you to place MATLAB command in a simple text file and then tell MATLAB to open the file and execute the commands exactly as it would if you typed them at the MATLAB Command Window.

M-Files must end with the extension ‘.m’. For example, homework1.m

There are many ways to load M-file Editor:

1. -Click on start → programs → MATLAB → R2017a → M-file Editor

2-From the MATLAB, chose New from the **Home** tab and select **Script**.

**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |

**Procedure:**

Matrix Operations:

To type a matrix into MATLAB you must:

* Begin with a square bracket **[**.
* Separate elements in a row with commas or spaces.
* Use a semicolon (**;**) to separate rows
* End the matrix with another square bracket].

Example-1:

a=[1,-2,0;10,-6,2;1,11,-9] or a=[1 -2 0; 10 -6 2; 1 11 -9]

Addition, Subtraction, Multiplication and Division:

Example-2:

A=[1 10 200 ; -50 0 -20 ; 44 25 60 ];

B=[1 15 -32 ; 14 20 20 ; 2 5 3];

C=[1 1 1 ; 25 -6 -6; 14 89 300];

A+B+C

A+B-C

A+B-C-10

A/C

A\*B\*C

2\*A

A^2 % A\*A

A.^2 % element wise square

Matrix Generator:

Example-3:

zeros(3,3)

ones(3,2)

rand(3,3)

Row Index and Column Index:

Example-4:

B=[0 20 90; 12 -34 45]

B(2,3)

B(1,2)

B(2,2)

Transpose, Determinant & Inverse of a Matrix:

Example-5:

W=[1 2 3; -4 -5 -6; 0 1 0]

W' %Transpose

det(W) %Determinant

inv(W) %Inverse

Control Statement

For Loop

Execute statements specified number of times

*For index = expression*

*Statement group x*

*End*

Where: **expression** is a matrix, usually the expression is something like **m:n** or **m:i:n**

where **m** is the beginning value, **n** the ending value and **i** is the increment.

Example-6:

The following statements will find the squares of all integers starting from

1 to10 with increment i = 2, the code would be

for i = 1:2:10

x = i^2

end

The output of this code: x = 1, 9,25,49,81.

Nested For Loop

Example-7:

m=3;

n=5;

for i=1:m

for j=1:n

f(i,j)=i;

end

end

f

The output of this code: f = 1 1 1 1 1

2 2 2 2 2

3 3 3 3 3

While Loop

Repeatedly execute statements while condition is true

Example-8:

i=1

d=0

while i<5 % i>1

d=i+1

i=d

end

disp('loop ended')

d

i

The output of this code: d=5, i = 5

Relation Operators with IF Else Statement:

MATLAB has six relational operators to make logical operations; these operators are shown in the following table

|  |  |
| --- | --- |
| Rational operator | description |
|  | |
| < | Less than. |
| <= | Less than or equal. |
| > | Greater than. |
| >= | Greater than or equal to. |
| = = | Equal to. |
| ~ = | Not equal to. |

Example-9:

a=5;

b=6;

if a~=b

disp('unequal')

else

disp('equal')

end

Logical Operators with IF Else Statement:

Symbol Meaning

& AND

| OR

˜ NOT

Example-10:

x=4; y=6;

if x<1 & y<1

z=0

elseif x>1 & y<1

z=1

elseif x>1 | y<1

z=2

end

The output of this code: z = 2

Solving a set of Linear Equations:

If a system of *n* linear equations in *n* unknowns can be expressed in form

**Ax = b**

Where A is an *n* x *n* matrix, and b is a column vector then, the solution to this system of equations can be expressed as

**x = A-1 b**

Where **A-1 is** the matrix inverse of A. In MATLAB, this system of equations can be accomplished by the left division (\).

Example-11:

Consider the following system of equations

3x+10y-z=0

-2x+y-10z=-2 x+y-z=3

This system of equations can be solved in MATLAB with the following code:

A=[3 10 -1;-2 1 -10; 1 1 -1];

b=[0;-2;3];

x=inv(A)\*b

x=A\b

or

[x,y,z]=solve('3\*x+10\*y-z=0','-2\*x+y-10\*z=-2','x+y-z=3')

The output of this code: x = 3.5000, -1.1111, -0.6111

**Report Question:**

1. Try the following commands for the matrix ‘W’

A=W(3,2)

B=W(1,1)

C=W(3,3)

D=W(2,2)

E=A+B

Ans: A =5, B =10, C =2, D =95, E =15, F =1.5850

1. Solve the following system of equations:

*17x1+2x2+1x3+5x4=4   
5x1+6x2+7x3+1x4=-1   
9x1-10x2+11x3+12x4=10   
13x1+14x2+15x3-9x4 = 6*

Ans: x1 = 0.5353, x2 = -0.9497, x3 = 0.3914, x4 = -0.7184

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha

**Experiment No. 3:**

**Name of the Experiment: Iterative Processes for Root Finding (Iterative Method, Aitken’s ∆² Acceleration Method, Bisection Method)**

**Objectives:**

1. Explain basic features configurations and applications of some signals in digital domain.
2. Familiar with the practical implementation of Iterative Method.
3. Familiar with the practical implementation of Aitken’s ∆² Acceleration Method.
4. Familiar with the practical implementation of Bisection Method.
5. Explain different digital systems and their properties.
6. Explain a system through difference equation in digital domain.
7. Manipulate frequency response of a system in digital domain.

**Learning Outcome:** After completing this experiment the students will be able to:

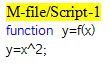
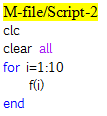
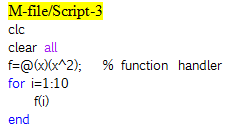
1. Define Middle Term Breaking Method
2. Define Fixed Point Iteration to Root Finding
3. Define Iterative Process to Locate the System Root.
4. Define Aitken’s ∆² Acceleration Method to Locate the System Root.
5. Define Bisection Process to Locate the System Root.

**Theory:**

**Iterative Process**

At first we have to introduce with MATLAB function declaration & function handler.

**Example-1:**



A formula can be developed for simple *fixed-point iteration* by arranging the function so that is on the left hand side of the equation:

(1)

Eq. (1) can be used to compute a new estimate as expressed by the iterative formula

(2)

The approximate error for this eq. (2) can be determined using the error estimator:

**Aitken's Method for Acceleration (Δ2Method):**

Here,

**Bisection method:**

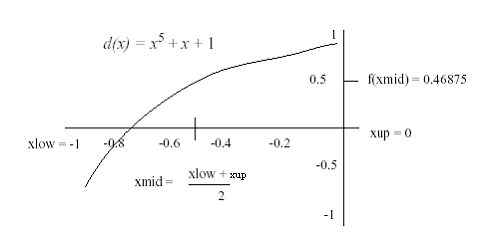
The Bisection method is one of the simplest procedures for finding root of a function in a given interval. The procedure is straightforward. The approximate location of the root is first determined by finding two values that bracket the root (a root is bracketed or enclosed if the function changes sign at the endpoints). Based on these a third value is calculated which is closer to the root than the original two value. A check is made to see if the new value is a root. Otherwise a new pair of bracket is generated from the three values, and the procedure is repeated.

Fig. 1: Bisection Method

Consider a function and let there be two values of ,  and (> ), bracketing a root of . The first step is to use the brackets  and  to generate a third value that is closer to the root. This new point is calculated as the mid-point between  and, namely . The method therefore gets its name from this bisecting of two values. It is also known as interval halving method. Test whether is a root of by evaluating the function at . If is not a root, then check if and have opposite signs i.e. .<0, root is in left half of interval. Or if and have same signs i.e. .>0, root is in right half of interval. Continue subdividing until interval width has been reduced to a size < tolerance. *Tips*: tolerance shall be 1x10**-4**

**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |
|  |  |

**Procedure:**

**Iterative Process Steps:**

1. The first step choice an initial approximation
2. Evaluate equation and determine the value of
3. Let
4. Check
5. If , then put and repeat steps (2) to (4)
6. Continue evaluating steps (2) to (4) until has been reduced to a value 1
7. If , then and stop evaluating iterations.

**Example:** Use simple fixed-point iteration to locate the root of ;

Solution: The function can be separated directly and expressed in the form of eq. (2) as

cos

**Example\_Ans:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0.000000000000000 |  |
| 1 | 1.000000000000000 | 100 |
| 2 | 0.540302305868140 | 85.081571768092545 |
| 3 | 0.857553215846393 | 36.994894790888438 |
| 4 | 0.654289790497779 | 31.066268846098438 |
| 5 | 0.793480358742566 | 17.541778660452646 |
| 6 | 0.701368773622757 | 13.133117495954123 |
| 7 | 0.763959682900654 | 8.192959743667133 |
| 8 | 0.722102425026708 | 5.796581817655344 |
| 9 | 0.750417761763761 | 3.773276457436351 |
| 10 | 0.731404042422510 | 2.599619121364808 |
| . | . | . |
| . | . | . |
| . | . | . |
| 42 | 0.739085108473799 | 8.317141693569484e-06 |

Thus, each iteration brings the estimate to the true value of the root: 0.739085108473799

On the 42th iteration.

**Aitken's Method for Acceleration (Δ2Method) Steps:**

1. The first step choice an initial approximation
2. Evaluate equation and determine the values ,
3. Determine the value of
4. Determine the value of
5. Determine the value of
6. Check
7. If , then put and repeat steps (2) to (6)
8. Continue evaluating steps (2) to (6) until accuracy has been reduced to a value 1
9. If , then Aitken’s and stop evaluating iterations.

**Example\_Ans:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0.000000000000000 |  |
| 1 | 1.000000000000000 | 0.685073357326045 |
| 2 | 0.540302305868140 | 0.738660156167714 |
| 3 | 0.857553215846393 | 0.739085106356719 |
| 4 | 0.654289790497779 | 0.739085133215161 |
| 5 | 0.793480358742566 |  |
| 6 | 0.701368773622757 |  |
| 7 | 0.763959682900654 |  |
| 8 | 0.722102425026708 |  |
| 9 | 0.750417761763761 |  |
| 10 | 0.731404042422510 |  |
| . | . |  |
| . | . |  |
| . | . |  |
| 42 | 0.739085108473799 |  |

Thus, Aitken’s Acceleration Process brings the estimate closer to the true value of the

root: 0.739085133215161 on the 4th iteration

**Bisection method Steps:**

1. The first step choice the initial approximations and
2. Determine the value of
3. Determine the values
4. Check if then change

OR if then change

1. Determine the value of
2. Check
3. If then change and repeat steps (2) to (6)
4. Continue evaluating steps (2) to (6) until difference has been reduced to a value 1
5. If then and stop evaluating iterations.

Bisection Fig 1\_Ans: -0.754821777343750

**Report Question:**

1. Write MATLAB Program of Iterative and Aitken’s Δ2 Method for the following systems:
2. Write MATLAB Program of BisectionMethod for the following systems:

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha

**Experiment No. 4:**

**Name of the Experiment: Solutions to Non-Linear Equations (Secant Method & Regula Falsi Method)**

**Objectives:**

1. Familiar with the practical implementation of Precision of Root Finding for a System.
2. Familiar with the practical implementation of Regula Falsi Method.
3. Familiar with the practical implementation of Newton Raphson Method.
4. Familiar with the practical implementation of Secant Method.
5. Explain a system through difference equation in digital domain.

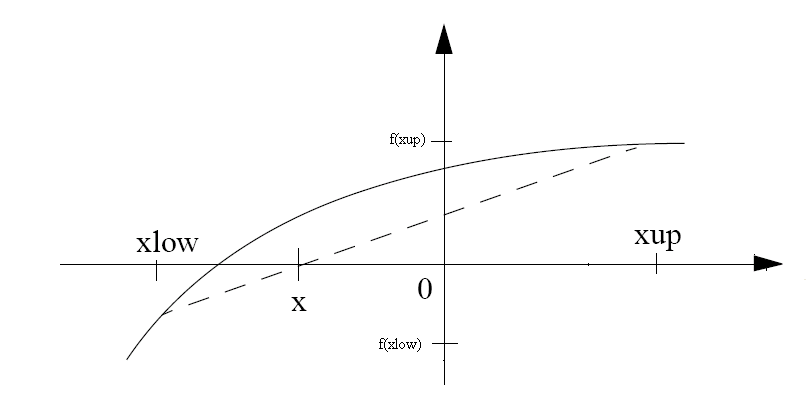
**Learning Outcome:** After completing this experiment the students will be able to:

1. Define Precision of Root Finding for a System.
2. Define Regula Falsi Method.
3. Define Newton Raphson Method.
4. Define Secant Method.

**Theory:**

**Regula Falsi Method**

A shortcoming of the bisection method is that, in dividing the interval from  to into equal halves, no account is taken of the magnitude of and. For example, if  is much closer to zero than, it is likely that the root is closer to  than to. An alternative method that exploits this graphical insight is to join and  by a straight line. The intersection of this line with the  axis represents an improved estimate of the root. The fact that the replacement of the curve by a straight line gives the false position of the root is the origin of the name, method of false position, or in Latin, Regula Falsi. It is also called the Linear Interpolation Method.



Using similar triangles, the intersection of the straight line with the  axis can be estimated as 

That is 

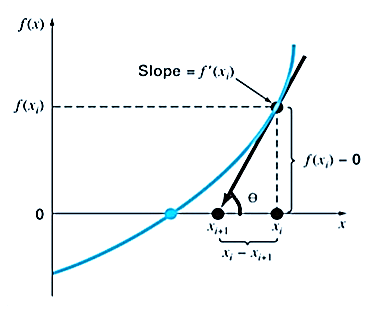
This is the False Position formulae. The value of then replaces whichever of the two initial guesses,  or, yields a function value with the same sign as. In this way, the values of  and always bracket the true root. The process is repeated until the root is estimated adequately.

**Newton-Raphson method**

Newton–Raphson method, named after [Isaac Newton](https://en.wikipedia.org/wiki/Isaac_Newton) and [Joseph Raphson](https://en.wikipedia.org/wiki/Joseph_Raphson), is a method for finding successively better approximations to the [roots](https://en.wikipedia.org/wiki/Root_of_a_function) (or zeroes) of a [real](https://en.wikipedia.org/wiki/Real_number)-valued [function](https://en.wikipedia.org/wiki/Function_(mathematics)). It is one example of a [root-finding algorithm](https://en.wikipedia.org/wiki/Root-finding_algorithm).

𝑥x : f ( x ) = 0 . {\displaystyle x:f(x)=0\,.}

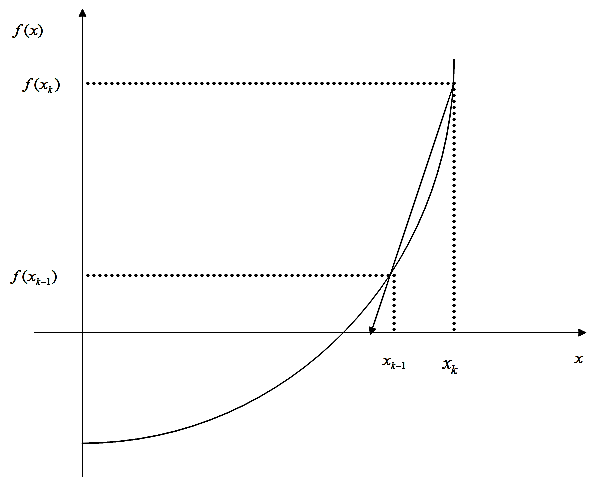
The Newton–Raphson method in one variable is implemented as follows:

The method starts with a function *f* defined over the [real numbers](https://en.wikipedia.org/wiki/Real_number) *x*, the function's [derivative](https://en.wikipedia.org/wiki/Derivative) *f ′*, and an initial guess *x*0 for a root of the function *f*. If the function satisfies the assumptions made in the derivation of the formula and the initial guess is close, then a better approximation *x*1 is

x 1 = x 0 − f ( x 0 ) f ′ ( x 0 ) . {\displaystyle x\_{1}=x\_{0}-{\frac {f(x\_{0})}{f'(x\_{0})}}\,.} Geometrically, (*x*1, 0) is the intersection of the *x*-axis and the [tangent](https://en.wikipedia.org/wiki/Tangent) of the [graph](https://en.wikipedia.org/wiki/Graph_of_a_function) of *f* at (*x*0, *f* (*x*0)).

The process is repeated as

x n + 1 = x n − f ( x n ) f ′ ( x n ) {\displaystyle x\_{n+1}=x\_{n}-{\frac {f(x\_{n})}{f'(x\_{n})}}\,}

Until a sufficiently accurate value is reached.

**The Secant Method:**

The secant method can be coded so that only one new function evaluation is required per iteration.   The formula for the secant method is the same one that was used in the Regula Falsi method, except that the logical decisions regarding how to define each succeeding term are different.

In the Secant method, the derivative can be approximated by a backward finite divided difference, as in the figure,



Using Newton-Raphson method,



Substituting,



Notice that the approach requires initial estimates of.

**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |
|  |  |

**Procedure:**

**Regula Falsi Method Steps:**

1. At first choice initial approximation (bigger one) and (smaller one)
2. Evaluate equation and determine the values
3. Determine the value
4. Check if then change

if then change

1. Find
2. If then repeat steps (2) to (5)
3. Continue evaluating steps (2) to (5) until difference has been reduced to a value 1
4. If then and stop evaluating iterations.

**Newton-Raphson method Steps:**

1. At first choice initial approximation
2. Evaluate equation and determine the values
3. Determine the value
4. Find
5. If then repeat steps (2) to (4)
6. Continue evaluating steps (2) to (4) until difference has been reduced to a value 1
7. If then and stop evaluating iterations.

**The Secant Method Steps:**

1. At first choice initial approximation (bigger one) and (smaller one)
2. Evaluate equation and determine the values
3. Determine the value
4. Find
5. If then repeat steps (2) to (4) and change
6. Continue evaluating steps (2) to (4) until difference has been reduced to a value 1
7. If then and stop evaluating iterations.

**Report Question:**

1. Write MATLAB Program of Regula Falsi, Secant, Newton-RaphsonMethods for the following systems: (here, for Regula Falsi and Secant; for Newton-Raphson)

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha

**Experiment No. 5:**

**Name of the Experiment: Interpolation by Newton-Gregory forward difference formula**

**Objectives:**

1. Familiar with the practical implementation of Precision of Root Finding for a System.
2. Familiar with the practical implementation of Interpolation by Newton-Gregory forward difference formula.
3. Familiar with the Merits and Demerits of Interpolation by Newton-Gregory forward difference formula.
4. Explain a system through difference equation in digital domain.

**Learning Outcome:** After completing this experiment the students will be able to:

1. Define Precision of Root Finding for a System.
2. Define Interpolation by Newton-Gregory forward difference formula.

**Theory:**

We are familiar with the analytical method of finding the derivative of a function when the functional relation between the dependent variable y and the independent variable x is known. However, in practice, most often functions are defined only by tabulated data, or the values of y for specified values of x can be found experimentally. Also in some cases, it is not possible to find the derivative of a function by analytical method. In such cases, the analytical process of differentiation breaks down and some numerical process have to be invented. The process of calculating the derivatives of a function by means of a set of given values of that function is called numerical differentiation. This process consists in replacing a complicated or an unknown function by an interpolation polynomial and then differentiating this polynomial as many times as desired.

**Forward Difference Formula**:

All numerical differentiation are done by expansion of Taylor series

…………..(1)

From (1)

……………..(2)

Where,  is the truncation error, which consists of terms containing h and higher order terms of h.

Total or True error = ………………..(3)

**Newton- Gregory Forward Difference Approach:**

Very often it so happens in practice that the given data set correspond to a sequence of equally spaced points. Here we can assume that

where is the starting point (sometimes, for convenience, the middle data point is taken as and in such a case the integer is allowed to take both negative and positive values.) and is the step size. ***Further it is enough to calculate simple differences rather than the divided differences as in the non-uniformly placed data set case***. These simple differences can be forward differences or backward differences . We will first look at forward differences and the interpolation polynomial based on forward differences.

The first order forward difference is defined as

|  |
| --- |
|  |

The second order forward difference is defined as

|  |
| --- |
|  |

The third order forward difference is defined as

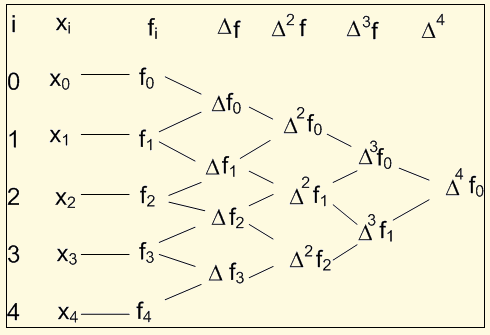
|  |
| --- |
|  |

The fourth order forward difference is defined as

|  |
| --- |
|  |

The kth order forward difference is defined as

This is known as Newton-Gregory forward difference interpolation polynomial. For convenience while constructing (10) one can first generate a forward difference table and use the from the table. Suppose we have data set then forward difference table looks as follows:

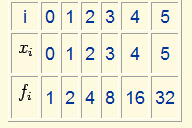


**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |
|  |  |

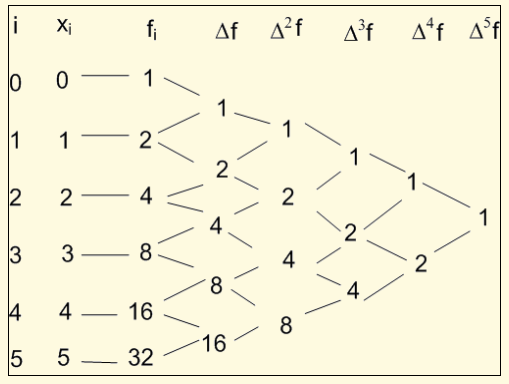
**Procedure:**

Given the following data, estimate using Newton-Gregory forward difference interpolation polynomial:

****

**Solution:**

Here we have six data points i.e. Let us first generate the forward difference table.

****

Here,

****

We have Newton-Gregory forward difference interpolation formula

**Steps:**

1. At first create initial approximation matrix
2. Evaluate the matrices as required to solve the equation
3. Write MATLAB expression of the formula with inputs
4. Determine the value using MATLAB Program

**Report Question:**

1. Write MATLAB Program of Newton-Gregory forward difference interpolation formula for the following systems
2. Ans. 17.9744
3. Ans. 45.0826

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha

**Experiment No. 6:**

**Name of the Experiment: Interpolation by Newton-Gregory backward difference formula**

**Objectives:**

1. Familiar with the practical implementation of Precision of Root Finding for a System.
2. Familiar with the practical implementation of Interpolation by Newton-Gregory backward difference formula.
3. Familiar with the Merits and Demerits of Interpolation by Newton-Gregory backward difference formula.
4. Explain a system through difference equation in digital domain.

**Learning Outcome:** After completing this experiment the students will be able to:

1. Define Precision of Root Finding for a System.
2. Define Interpolation by Newton-Gregory backward difference formula.

**Theory:**

We are familiar with the analytical method of finding the derivative of a function when the functional relation between the dependent variable y and the independent variable x is known. However, in practice, most often functions are defined only by tabulated data, or the values of y for specified values of x can be found experimentally. Also in some cases, it is not possible to find the derivative of a function by analytical method. In such cases, the analytical process of differentiation breaks down and some numerical process have to be invented. The process of calculating the derivatives of a function by means of a set of given values of that function is called numerical differentiation. This process consists in replacing a complicated or an unknown function by an interpolation polynomial and then differentiating this polynomial as many times as desired.

**Backward Difference Formula**:

All numerical differentiation are done by expansion of Taylor series

…………..(1)

From (1)

……………..(2)

Where,  is the truncation error, which consists of terms containing h and higher order terms of h.

Total or True error = ………………..(3)

**Newton- Gregory Backward Difference Approach:**

The first order backward difference is defined as

|  |
| --- |
|  |

The second order backward difference is defined as

|  |
| --- |
|  |

The third order backward difference is defined as

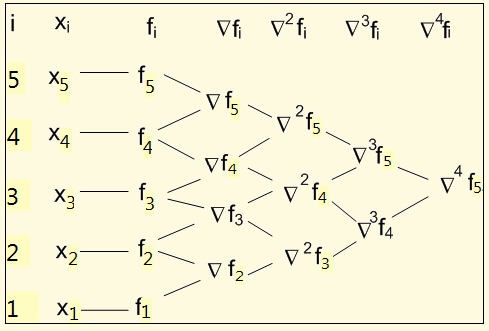
|  |
| --- |
|  |

The fourth order backward difference is defined as

|  |
| --- |
|  |

The kth order backward difference is defined as

|  |
| --- |
| This is known as Newton-Gregory backward difference interpolation polynomial. For convenience while constructing (10) one can first generate a backward difference table and use the from the table. Suppose we have data set then backward difference table looks as follows: |

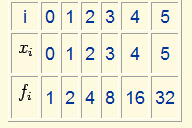
****

**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |
|  |  |

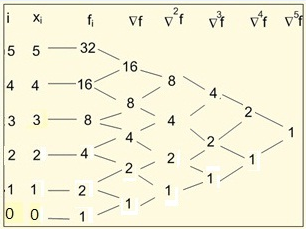
**Procedure:**

Given the following data, estimate using Newton-Gregory backward difference interpolation polynomial:

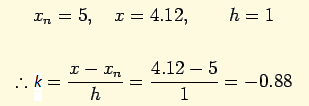


**Solution:**

Here we have six data points i.e. Let us first generate the backward difference table (next page).

****

Here,



We have Newton-Gregory backward difference interpolation formula

**Steps:**

1. At first create initial approximation matrix
2. Evaluate the matrices as required to solve the equation
3. Write MATLAB expression of the formula with inputs
4. Determine the value using MATLAB Program

**Report Question:**

1. Write MATLAB Program of Newton-Gregory backward difference interpolation formula for the following systems
2. Ans. 17.9744
3. Ans. 7.1914

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha

**Experiment No. 7:**

**Name of the Experiment: Numerical Differentiation**

**(Based on Newton-Gregory Forward & Backward Differences)**

**Objectives:**

1. Familiar with the practical implementation of numerical differentiation for a System.
2. Familiar with the practical implementation of numerical differentiation by Newton-Gregory backward difference formula.
3. Familiar with the merits and demerits of numerical differentiation by Newton-Gregory backward difference formula.
4. Explain a system through difference equation in digital domain.

**Learning Outcome:** After completing this experiment the students will be able to:

1. Define Precision of numerical differentiation for a System.
2. Define numerical differentiation by Newton-Gregory backward difference formula.

**Theory:**

We are familiar with the analytical method of finding the derivative of a function when the functional relation between the dependent variable y and the independent variable x is known. However, in practice, most often functions are defined only by tabulated data, or the values of y for specified values of x can be found experimentally. Also in some cases, it is not possible to find the derivative of a function by analytical method. In such cases, the analytical process of differentiation breaks down and some numerical process have to be invented. The process of calculating the derivatives of a function by means of a set of given values of that function is called numerical differentiation. This process consists in replacing a complicated or an unknown function by an interpolation polynomial and then differentiating this polynomial as many times as desired.

**Forward Difference Formula**:

All numerical differentiation are done by expansion of Taylor series

…………..(1)

From (1)

……………..(2)

Where,  is the truncation error, which consists of terms containing h and higher order terms of h.

Total or True error = ………………..(3)

**Newton- Gregory Forward Difference Approach:**

Very often it so happens in practice that the given data set correspond to a sequence of equally spaced points. Here we can assume that

where is the starting point (sometimes, for convenience, the middle data point is taken as and in such a case the integer is allowed to take both negative and positive values.) and is the step size. ***Further it is enough to calculate simple differences rather than the divided differences as in the non-uniformly placed data set case***. These simple differences can be forward differences or backward differences . We will first look at forward differences and the interpolation polynomial based on forward differences.

The first order forward difference is defined as

|  |
| --- |
|  |

The second order forward difference is defined as

|  |
| --- |
|  |

The third order forward difference is defined as

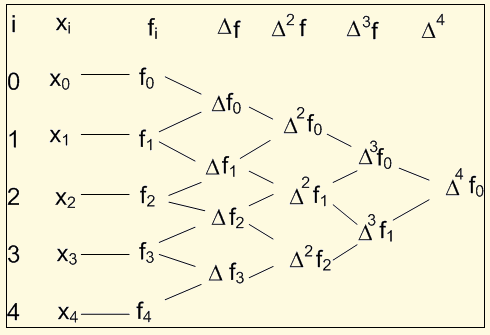
|  |
| --- |
|  |

The fourth order forward difference is defined as

|  |
| --- |
|  |

The kth order forward difference is defined as

This is known as Newton-Gregory forward difference interpolation polynomial. For convenience while constructing (10) one can first generate a forward difference table and use the from the table. Suppose we have data set then forward difference table looks as follows:

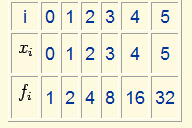
****

**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |
|  |  |

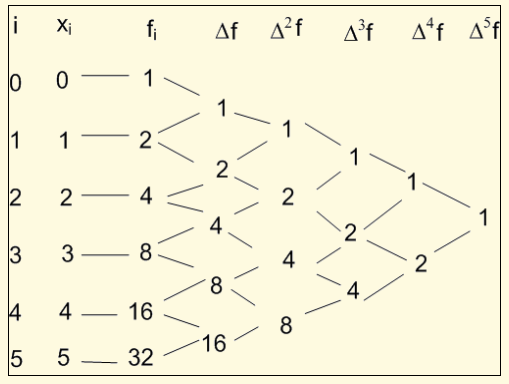
**Procedure:**

Given the following data, estimate using Newton-Gregory forward difference interpolation polynomial:



**Solution:**

Here we have six data points i.e. Let us first generate the forward difference table.

****

Here,

****

We have Newton-Gregory forward difference interpolation formula

Now, [ From, "Numerical Methods" by S.Balachandra Rao, Equation - 5.5 ]

(Ans.)

Similarly for Newton-Gregory backward difference interpolation formula

**Steps:**

1. At first create initial approximation matrix
2. Evaluate the matrices as required to solve the equation
3. Write MATLAB expression of the formula with inputs
4. Determine the value using MATLAB Program

**Report Question:**

1. Write MATLAB Program of Numerical Differentiation for Newton-Gregory forward & backward difference interpolation formula for the following systems
2. Ans. 8.24 (forward), 14.24 (backward)
3. Ans. 26.038933 (forward), 34.31893333 (backward)

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha

**Experiment No. 8:**

**Name of the Experiment: Numerical Differentiation (Based on Lagrange Interpolation) & Numerical Integration (Based on Simple Trapezium Rule)**

**Objectives:**

1. Familiar with the practical implementation of numerical differentiation and numerical integration for a System.
2. Familiar with the practical implementation of numerical differentiation based on Lagrange Interpolation for a System.
3. Familiar with the practical implementation of numerical integration based on Simple Trapezium Rule for a System.
4. Explain a system through difference equation in digital domain.

**Learning Outcome:** After completing this experiment the students will be able to:

1. Define Precision of numerical differentiation and numerical integration for a System.
2. Define numerical differentiation based on Lagrange Interpolation formula.
3. Define numerical integration based on Simple Trapezium Rule formula.

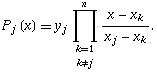
**Theory:**

1. **Lagrange Polynomial:**

The Lagrange interpolating polynomial is the [polynomial](http://mathworld.wolfram.com/Polynomial.html)  of degree that passes through the  points (x_1,y_1==f(x_1)), (x_2,y_2==f(x_2)), ..., (x_n,y_n==f(x_n)), and is given by

P(x)==sum_(j==1)^nP_j(x),

where



Written explicitly,

|  |  |  |
| --- | --- | --- |
| P(x) | = | ((x-x_2)(x-x_3)...(x-x_n))/((x_1-x_2)(x_1-x_3)...(x_1-x_n))y_1+((x-x_1)(x-x_3)...(x-x_n))/((x_2-x_1)(x_2-x_3)...(x_2-x_n))y_2+...+((x-x_1)(x-x_2)...(x-x_(n-1)))/((x_n-x_1)(x_n-x_2)...(x_n-x_(n-1)))y_n. |

When constructing interpolating polynomials, there is a tradeoff between having a better fit and having a smooth well-behaved fitting function. The more data points that are used in the interpolation, the higher the degree of the resulting polynomial, and therefore the greater oscillation it will exhibit between the data points. Therefore, a high-degree interpolation may be a poor predictor of the function between points, although the accuracy at the data points will be "perfect."

For n==3 points,

|  |  |  |
| --- | --- | --- |
| P(x) | = | ((x-x_2)(x-x_3))/((x_1-x_2)(x_1-x_3))y_1+((x-x_1)(x-x_3))/((x_2-x_1)(x_2-x_3))y_2+((x-x_1)(x-x_2))/((x_3-x_1)(x_3-x_2))y_3 |

Note that the function  passes through the points (x_i,y_i), as can be seen for the case n==3,

|  |  |  |
| --- | --- | --- |
| P(x_1) | = | ((x_1-x_2)(x_1-x_3))/((x_1-x_2)(x_1-x_3))y_1+((x_1-x_1)(x_1-x_3))/((x_2-x_1)(x_2-x_3))y_2+((x_1-x_1)(x_1-x_2))/((x_3-x_1)(x_3-x_2))y_3==y_1 |
| P(x_2) | = | ((x_2-x_2)(x_2-x_3))/((x_1-x_2)(x_1-x_3))y_1+((x_2-x_1)(x_2-x_3))/((x_2-x_1)(x_2-x_3))y_2+((x_2-x_1)(x_2-x_2))/((x_3-x_1)(x_3-x_2))y_3==y_2 |
| P(x_3) | = | ((x_3-x_2)(x_3-x_3))/((x_1-x_2)(x_1-x_3))y_1+((x_3-x_1)(x_3-x_3))/((x_2-x_1)(x_2-x_3))y_2+((x_3-x_1)(x_3-x_2))/((x_3-x_1)(x_3-x_2))y_3==y_3. |

1. **Simple Trapezium Rule:**

We know that higher order differences are negligible for small in the case of well behaved functions.

E B

Y

A

O π/4 X

One Strip

Two Strip

The above formula is called the Trapezium Rule.

**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |
|  |  |

**Procedure:**

1. **Find from the following data using Lagrange Polynomial Interpolation:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x 0.05 0.10 0.20 0.26 | | | | | | |
|  | 0.05000 | 0.0999 | 0.1987 | 0.2571 |  |  |

**Solution:**

Using Lagrange Polynomial Interpolation we get,

Differentiating with respect to we have

When , we get

**Steps for the Lagrange Polynomial:**

For a given *x* and f(x), two sets of *(N+1)* data pairs, (*xi , fi*), *i*= *0, 1, . ….. N:*

Set SUM=0

DO FOR *i*=0 to *N*

Set *Y*=1

DO FOR *j*=0 to *N*

IF *j~=i*

Set *Y=Y\*(x-x(j))/(x(i)-x(j))*

End DO(*j*)

*Y=Y\*f*

*SUM=SUM+Y*

1. **Evaluate Numerically using Simple Trapezium Rule**

Table 2

|  |  |
| --- | --- |
| X |  |
| 0 | 1.000000 |
| 0.1 | 0.990050 |
| 0.2 | 0.960789 |
| 0.3 | 0.913931 |
| 0.4 | 0.852144 |
| 0.5 | 0.778801 |
| 0.6 | 0.697676 |
| 0.7 | 0.612626 |
| 0.8 | 0.527292 |

From **Simple Trapezium Rule:**

We know that higher order differences are negligible for small in the case of well-behaved functions.

Here, for one strip

for two strip

for three strip

for four strip

For one strip (i.e. h=0.8),

For two strip (i.e. h=0.4),

For three strip (i.e. h=0.2),

For four strip (i.e. h=0.1),

**Steps:**

1. At first evaluate initial approximation matrices
2. Evaluate the matrices as required to solve the equation
3. Write MATLAB expression of the formula , ,

**Report Question:**

1. Write MATLAB Program of Numerical Differentiation for Lagrange Polynomial interpolation formula for the following systems:
2. Ans. 0.24
3. Ans. 1.24
4. Write MATLAB Program of Numerical Integration using Simple Trapezium Rule for up to four strip.

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha

**Experiment No. 9:**

**Name of the Experiment: Numerical Integration by Romberg Integration and Simpson's 1/3rd, 3/8th Rule**

**Objectives:**

1. Familiar with the practical implementation of numerical integration for a System.
2. Familiar with the practical implementation of numerical integration based on Romberg Integration for a System.
3. Familiar with the practical implementation of numerical integration based on Simpson's 1/3rd rule for a System.
4. Familiar with the practical implementation of numerical integration based on Simpson's 3/8th rule for a System.
5. Explain a system through difference features in digital domain.

**Learning Outcome:** After completing this experiment the students will be able to:

1. Define Precision of numerical integration for a System.
2. Define numerical integration based on Romberg Integration formula.
3. Define numerical integration based on Simpson's 1/3rd rule formula.
4. Define numerical integration based on Simpson's 3/8th rule formula.

**Theory:**

There are two cases in which engineers and scientists may require the help of numerical integration technique. (1) Where experimental data is obtained whose integral may be required and (2) where a closed form formula for integrating a function using calculus is difficult or so complicated as to be almost useless. For example the integral



Since there is no analytic expression for , numerical integration technique must be used to obtain approximate values of .

Formulae for numerical integration called quadrature are based on fitting a polynomial through a specified set of points (experimental data or function values of the complicated function) and integrating (finding the area under the fitted polynomial) this approximating function. Any one of the interpolation polynomials studied earlier may be used.

1. **Romberg Integration**

In the case of Trapezoidal rule:

In the case of Simpson’s rule:

1. **Simpson’s 1/3 Rule**

This is based on approximating the function f(x) by fitting **quadratics** through sets of **three** points. For only three points it can be written as:



This is called second-degree Newton-Cotes formula.

It is evident that the result of integration between x1 and x1+nh can be written as

**

In using the above formula it is implied that f is known at an **odd number of points** (**n+1 is odd,** where n is the no. of subintervals).

1. **Simpson’s 3/8 Rule**

This is based on approximating the function f(x) by fitting **cubic** interpolating polynomial through sets of **four** points. For only four points it can be written as:



This is called third-degree Newton-Cotes formula.

It is evident that the result of integration between x1 and x1+nh can be written as



In using the above formula it is implied that f is known at (n+1) points where **n is divisible by 3**.

**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |

**Procedure:**

1. **Romberg Integration**

Using the values given in the following Table 2, find by Romberg’s integration.

|  |  |
| --- | --- |
| X |  |
| 0 | 1.000000 |
| 0.1 | 0.990050 |
| 0.2 | 0.960789 |
| 0.3 | 0.913931 |
| 0.4 | 0.852144 |
| 0.5 | 0.778801 |
| 0.6 | 0.697676 |
| 0.7 | 0.612626 |
| 0.8 | 0.527292 |

**Solution:**Table 1

From Simple Trapezium Rule, we know that higher order differences are negligible for small in the case of well behaved functions.

Here, for one strip

for two strip

for three strip

for four strip

For one strip (i.e. h=0.8),

For two strip (i.e. h=0.4),

For three strip (i.e. h=0.2),

For four strip (i.e. h=0.1),

Table 2

|  |  |
| --- | --- |
| h |  |
| 0.8 | 0.6109168 |
| 0.4 | 0.6463160 |
| 0.2 | = 0.6548510 |
| 0.1 | = 0.6569663 |
|  |  |

Now From Romberg’s Formula:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Ans.:** = 0.6576698

**Steps:**

1. At first evaluate initial approximation matrices
2. Evaluate corresponding and matrices as required to solve the equation
3. Write MATLAB expression of the formula , ,
4. Determine , , using MATLAB.
5. **Simpson’s 1/3 Rule:**

Evaluate by using Simpson’s composite 1/3 rule.

**Solutiuon:** Compare the values with the exact value of the integral

**Table 1**

|  |  |
| --- | --- |
| X |  |
| 1 | 1 |
| 2 | 0.5 |
| 3 | 0.33 |
| 4 | 0.25 |
| 5 | 0.2 |
| 6 | 0.17 |
| 7 | 0.14 |

**Ans.:** (a) 1.96, Exact=1.9459101

1. **Simpson’s 3/8 Rule:**

Evaluate by using Simpson’s composite 3/8 rule.

**Solutiuon:** Compare the values with the exact value of the integral

**Table 2**

|  |  |
| --- | --- |
| X |  |
| 1 | 1 |
| 2 | 0.5 |
| 3 | 0.33 |
| 4 | 0.25 |
| 5 | 0.2 |
| 6 | 0.17 |
| 7 | 0.14 |

**Ans.:** 1.965, Exact=1.9459101

**Steps:**

An algorithm for integrating a **tabulated function** using composite trakpezoidal rule:

Remarks: f1, f2,………, fn+1 are the tabulated values at x1, x1+h,………x1+nh (n+1 points)

1 Read

2 n=length(x)

3

4 Read

5

6

7

8

9

10

**Report Question:**

1. Write MATLAB Program of Numerical Integration using Romberg Integration for up to four strip.
2. Write MATLAB Program of Numerical Differentiation for Simpson’s composite 1/3 & 3/8 rules for the following systems:

**Reference Book:** Applied Numerical Methods with MATLAB (for Engineers & Scientists) (Special Indian Edition), Steven C. Chapra

**Experiment No. 10:**

**Name of the Experiment: Solution of system of Linear Equations By a direct method (LU Decomposition) and an indirect method (Jacobi or Gauss Seidel Method)**

**Objectives:**

1. Familiar with the practical implementation of the different techniques of finding solution of a set of n linear algebraic equations in n unknowns.
2. Familiar with the practical implementation of LU Decomposition for system solution of linear equations.
3. Familiar with the practical implementation of Jacobi or Gauss Seidel method for system solution of linear equations.
4. Explain a system through difference features in digital domain.

**Learning Outcome:** After completing this experiment the students will be able to:

1. Define Precision of system solution for linear equations.
2. Define LU Decomposition for system solution of linear equations.
3. Define Jacobi or Gauss Seidel method for system solution of linear equations.

**Theory:**

**Concept of linear equations and their solution**

A set of linear algebraic equations looks like this:



 (1)

… … … …



Here the N unknowns xj , j = 1, 2, . . .,N are related by M equations. The coefficients aij with i = 1, 2, . . .,M and j = 1, 2, . . .,N are known numbers, as are the right-hand side quantities bi, i = 1, 2, . . .,M.

# Existence of solution

If N = M then there are as many equations as unknowns, and there is a good chance of solving for a unique solution set of xj’s. Analytically, there can fail to be a unique solution if one or more of the M equations is a linear combination of the others (This condition is called row degeneracy), or if all equations contain certain variables only in exactly the same linear combination(This is called column degeneracy). (For square matrices, a row degeneracy implies a column degeneracy, and vice versa.) A set of equations that is degenerate is called singular.

Numerically, at least two additional things can go wrong:

• While not exact linear combinations of each other, some of the equations may be so close to linearly dependent that round off errors in the machine renders them linearly dependent at some stage in the solution process. In this case your numerical procedure will fail, and it can tell you that it has failed.

• Accumulated round off errors in the solution process can swamp the true solution. This problem particularly emerges if N is too large. The numerical procedure does not fail algorithmically. However, it returns a set of x’s that are wrong, as can be discovered by direct substitution back into the original equations. The closer a set of equations is to being singular, the more likely this is to happen.

## Matrices

Equation (1) can be written in matrix form as

**A** · **x** = **b** (2)

Here the raised dot denotes matrix multiplication, **A** is the matrix of coefficients, **x** is the column vector of unknowns and **b** is the right-hand side written as a column vector,

## Finding Solution

There are so many ways to solve this set of equations. Below are some important methods.

**(1) Using the backslash and pseudo-inverse operator**

In MATLAB, the easiest way to determine whether Ax = b has a solution, and to find such a solution when it does, is to use the backslash operator. Exactly what returns is a bit complicated to describe, but if there is a solution to **A** · **x** = **b**, then returns one. Warnings: (1) returns a result in many cases when there is no solution to **A** · **x** = **b**. (2) sometimes causes a warning to be issued, even when it returns a solution. This means that you can't just use the backslash operator: you have to check that what it returns is a solution. (In any case, it's just good common sense to check numerical computations as you do them.) In MATLAB this can be done as follows:

Using backslash operator:

x = A\b;

You can also use the pseudo-inverse operator:

x=pinv(A)\*b; % it is also guaranteed to solve Ax = b, if Ax = b has a solution.

As with the backslash operator, you have to check the result.

**(2) Using Gauss-Jordan Elimination and Pivoting**

To illustrate the method let us consider three equations with three unknowns:

 (A)

 (B)

 (C)

Here the quantities bi, i = 1, 2, . . .,M’s are replaced by aiN+1, where i=1,2, ….M for simplicity of understanding the algorithm.

The First Step is to eliminate the first term from Equations (B) and (C). (Dividing (A) by a11 and multiplying by a21 and subtracting from (B) eliminates x1 from (B) as shown below)



Let, , then



Similarly multiplying equation (A) by and subtracting from (C), we get



Observe that and are both zero.

In the steps above it is assumed that a11 is not zero. This case will be considered later in this experiment.

The above elimination procedure is called triangularization.

**Procedure:**

**For triangularizing n equations in n unknowns:**

1 

2 

3 

4 

5 

6 





The reduced equations are:







The next step is to eliminate from the third equation. This is done by multiplying second equation by and subtracting the resulting equation from the third. So, same algorithm can be used.

Finally the equations will take the form:







The above set of equations are said to be in triangular (Upper) form.

From the above upper triangular form of equations, the values of unknowns can be obtained by back substitution as follows:







Algorithmically, the back substitution for n unknowns is shown below:

1 

2 

3 

4 

5 

6 



**Pivoting**

In the triangularization algorithm we have used,



Here it is assumed that is not zero. If it happens to be zero or nearly zero, the algorithm will lead to no results or meaningless results. If any of the is small it would be necessary to reorder the equations. It is noted that the value of would be modified during the elimination process and there is no way of predicting their values at the start of the procedure.

The elements are called pivot elements. In the elimination procedure the pivot should not be zero or a small number. In fact for maximum precision the pivot element should be the largest in absolute value of all the elements below it in its column, i.e. should be picked up as the maximum of all where, 

So, during the Gauss elimination, elements should be searched and the equation with the maximum value of should be interchanged with the current position. For example if during elimination we have the following situation:



As  2nd and 3rd equations should be interchanged to yield:



It should be noted that interchange of equations does not affect the solution.

The algorithm for picking the largest element as the pivot and interchanging the equations is called pivotal condensation.

**Procedure:**

**For pivotal condensation**

1 

2 

3 

4 

5 

6 

7 



8 

9 

10 

11 

12 





**(3) Using Gauss-Seidel Iterative Method**

There are several iterative methods for the solution of linear systems. One of the efficient iterative methods is the Gauss-Seidel method.

Let us consider the system of equations:



The Gauss-Seidel iterative process is suggested by the following equations:



The very first iteration, that is  (for n equations) are set equal to zero and  is calculated. The main point of Gauss-Seidel iterative process to observe is that always the latest approximations for the values of variables are used in an iteration step.

It is to be noted that in some cases the iteration diverges rather than it converges. Both the divergence and convergence can occur even with the same set of equations but with the change in the order. The sufficient condition for the Gauss-Seidel iteration to converge is stated below.

The Gauss-Seidel iteration for the solution will converge (if there is any solution) if the matrix  (as defined previously) is strictly diagonally dominant matrix.

A matrix  of dimension  is said to be strictly diagonally dominant provided that



**List of Equipment:**

|  |  |
| --- | --- |
| 1. | Desktop PC |
| 2. | Software MATLAB R2017a |

**Report Question:**

1. Given the simultaneous equations shown below (i) triangularize them (ii) use back substitution to solve for , , .



For generalization, you will have to write a program for triangularizing n equations in n unknowns with back substitution.

1. Modify the MATLAB program written in exercise 1 to include pivotal condensation.
2. Try to solve the following systems of equations (i) Gauss-Jordan elimination (ii) Gauss-Jordan elimination with pivoting

(A)  (B) 

(C) 

1. Solve the following equations using Gauss-Seidel iteration process:

(A)  (B) 

(C)  (D) 

**Reference Book:**

1. Applied Numerical Methods with MATLAB (for Engineers & Scientists)

(Special Indian Edition),

Steven C. Chapra

1. Numerical Methods,

S. Balachandra Rao, C.K.Shantha