**RAMANUJAN COLLEGE**

**(UNIVERSITY OF DELHI)**

**2022-2023**

**(GE-IV )**

**Numerical Methods Practical File**

**Submitted By: Submitted To:**

Prakhar KhugshalMr. Sainyam Gupta

**Roll No. =** 20211441

**B.Sc (Hons) Computer Science**

**Semester IV**

**INDEX**

|  |  |
| --- | --- |
| **S.no** | **Title** |
| **1.** | Bisection method |
| **2.(a)**  **(b)** | Secant method |
| Regula Falsi method |
| **3.** | Newton Raphson method |
| **4.(a)**  **(b)** | Gauss elimination method |
| Gauss Jordan method |
| **5.(a)**  **(b)** | Gauss Jacobi method |
| Gauss seidel method |
| **6.(a)**  **(b)** | Lagrange interpolation |
| Newton divided difference interpolation |
| **7.(a)**  **(b)** | Trapezoidal method |
| Simpson 1/3 method |
| **8.** | Euler’s method |

Practical 1 Prakhar Khugshal

20211441 B.Sc(H) Computer Science 4th Semester

Bisection Method

**Question 1**

**x0** = **1.0; x1**=**2.0; NMax**=**20; eps**=**0.0001;**

**f**[**x**\_]**:**=**Cos**[**x**]**;**

**If**[**N**[**f**[**x0**] \***f**[**x1**]] >**0, Print**[

**"Yours values do opt staisfy the IVP so change the value."**]**, For**[**i**=**1, i**≤**NMax, i**++**, m**= (**x0**+**x1**) /**2;**

**If**[**Abs**[(**x1**-**x0**) /**2**] <**eps, Return**[**m**]**, Print**[**i, "th iteration value is :", m**]**; Print**[**"Estimated error in ",**

**i, "th iteration is:",**(**x1**-**x0**) /**2**]

**If**[**f**[**m**] \***f**[**x1**] >**0, x1**=**m, x0**=**m**]]]**;**

**Print**[**"Estimated error in", i, "th iteration is:",**(**x1**-**x0**) /**2**]]

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}**, PlotRange**→{-**1, 1**}**,**

**PlotStyle**→**Red, PlotLabel**→**"f**[**x**]=**" f**[**x**]**, AxesLabel**→{**x, f**[**x**]}]

1th iteration value is :1.5

Estimated error in 1th iteration is:0.5 2th iteration value is :1.75

Estimated error in 2th iteration is:0.25 3th iteration value is :1.625

Estimated error in 3th iteration is:0.125 4th iteration value is :1.5625

Estimated error in 4th iteration is:0.0625 5th iteration value is :1.59375

Estimated error in 5th iteration is:0.03125 6th iteration value is :1.57813

Estimated error in 6th iteration is:0.015625 7th iteration value is :1.57031

Estimated error in 7th iteration is:0.0078125 8th iteration value is :1.57422

Estimated error in 8th iteration is:0.00390625 9th iteration value is :1.57227

Estimated error in 9th iteration is:0.00195313 10th iteration value is :1.57129

Estimated error in 10th iteration is:0.000976563 11th iteration value is :1.5708

Estimated error in 11th iteration is:0.000488281 12th iteration value is :1.57056

Estimated error in 12th iteration is:0.000244141 13th iteration value is :1.57068

Estimated error in 13th iteration is:0.00012207

Return[1.57074]

f[x]=cos(x)

cos(x) 1.0

-

|  |  |
| --- | --- |
| 0.5 |  |
| 1  -0.5  -1.0 | 1 2 3 |

x

Question 2

**x0**=**0; x1**=**1.0; NMax**=**20; eps**=**0.0001;**

**f**[**x**\_]**:**=**x^3**-**5 x**+**1; If**[**N**[**f**[**x0**] \***f**[**x1**]] >**0, Print**[

**"Yours values do opt staisfy the IVP so change the value."**]**, For**[**i**=**1, i**≤**NMax, i**++**, m**= (**x0**+**x1**) /**2;**

**If**[**Abs**[(**x1**-**x0**) /**2**] <**eps, Return**[**m**]**, Print**[**i, "th iteration value is :", m**]**; Print**[**"Estimated error in ",**

**i, "th iteration is:",**(**x1**-**x0**) /**2**]

**If**[**f**[**m**] \***f**[**x1**] >**0, x1**=**m, x0**=**m**]]]**;**

**Print**[**"Estimated error in", i, "th iteration is:",**(**x1**-**x0**) /**2**]]

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}**, PlotRange**→{-**1, 1**}**,**

**PlotStyle**→**Red, PlotLabel**→**"f**[**x**]=**" f**[**x**]**, AxesLabel**→{**x, f**[**x**]}]

1th iteration value is :0.5

Estimated error in 1th iteration is:0.5 2th iteration value is :0.25

Estimated error in 2th iteration is:0.25 3th iteration value is :0.125

Estimated error in 3th iteration is:0.125 4th iteration value is :0.1875

Estimated error in 4th iteration is:0.0625 5th iteration value is :0.21875

Estimated error in 5th iteration is:0.03125 6th iteration value is :0.203125

Estimated error in 6th iteration is:0.015625 7th iteration value is :0.195313

Estimated error in 7th iteration is:0.0078125 8th iteration value is :0.199219

Estimated error in 8th iteration is:0.00390625 9th iteration value is :0.201172

Estimated error in 9th iteration is:0.00195313 10th iteration value is :0.202148

Estimated error in 10th iteration is:0.000976563 11th iteration value is :0.20166

Estimated error in 11th iteration is:0.000488281 12th iteration value is :0.201416

Estimated error in 12th iteration is:0.000244141 13th iteration value is :0.201538

Estimated error in 13th iteration is:0.00012207

Return[0.201599]

f[x]=x 3 -5x+1

x3 -5x+1 1.0

-

|  |  |
| --- | --- |
| 0.5 |  |
| 1  -0.5  -1.0 | 1 2 3 |

x

Question 3 :

**x0** = **0; x1**=**1.0; NMax**=**20; eps**=**0.0001;**

**f**[**x**\_]**:**=**Cos**[**x**] -**x**\***Exp**[**x**]**;**

**If**[**N**[**f**[**x0**] \***f**[**x1**]] >**0, Print**[

**"Yours values do opt staisfy the IVP so change the value."**]**, For**[**i**=**1, i**≤**NMax, i**++**, m**= (**x0**+**x1**) /**2;**

**If**[**Abs**[(**x1**-**x0**) /**2**] <**eps, Return**[**m**]**, Print**[**i, "th iteration value is :", m**]**; Print**[**"Estimated error in ",**

**i, "th iteration is:",**(**x1**-**x0**) /**2**]

**If**[**f**[**m**] \***f**[**x1**] >**0, x1**=**m, x0**=**m**]]]**;**

**Print**[**"Estimated error in", i, "th iteration is:",**(**x1**-**x0**) /**2**]]

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}**, PlotRange**→{-**10, 10**}**,**

**PlotStyle**→**Red, PlotLabel**→**"f**[**x**]=**" f**[**x**]**, AxesLabel**→{**x, f**[**x**]}]

1th iteration value is :0.5

Estimated error in 1th iteration is:0.5 2th iteration value is :0.75

Estimated error in 2th iteration is:0.25 3th iteration value is :0.625

Estimated error in 3th iteration is:0.125 4th iteration value is :0.5625

Estimated error in 4th iteration is:0.0625 5th iteration value is :0.53125

Estimated error in 5th iteration is:0.03125 6th iteration value is :0.515625

Estimated error in 6th iteration is:0.015625 7th iteration value is :0.523438

Estimated error in 7th iteration is:0.0078125 8th iteration value is :0.519531

Estimated error in 8th iteration is:0.00390625 9th iteration value is :0.517578

Estimated error in 9th iteration is:0.00195313 10th iteration value is :0.518555

Estimated error in 10th iteration is:0.000976563 11th iteration value is :0.518066

Estimated error in 11th iteration is:0.000488281 12th iteration value is :0.517822

Estimated error in 12th iteration is:0.000244141 13th iteration value is :0.5177

Estimated error in 13th iteration is:0.00012207

Return[0.517761]

cos(x) -ⅇ x x

10

|  |  |
| --- | --- |
| 5 |  |
| 1  -5  -10 | 1 2 3 |

f[x]=cos(x) -ⅇ x x

x

-

Practical

2(a) --> Secant Method

Prakhar Khugshal || 20211441 || B Sc Hon Computer science ||

**x0**=**Input**[**"Enter first guess: "**]**; x1**=**Input**[**"Enter scond guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"x1**=**", x1**]**; Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**Cos**[**x**]**; Print**[**"f**[**x**]**:**=**", f**[**x**]] **For****i**=**1, i**≤**Nmax, i**++**,**

**x2**=**N****x1**-**f**[**x**] /**. x**→**x1**\***x1**-**x0**  **f**[**x**] /**. x**→**x1**-**f**[**x**] /**. x**→**x0****; If**[**Abs**[**x1**-**x2**] <**eps, Return**[**x2**]**, x0**=**x1; x1**=**x2**]**;**

**Print**[**"In", i, "th number of iterations the root is :", x2**]**;**

**Print**[**"estimated error is: ", Abs**[**x1**-**x0**]]**; Print**[**"root is : ", x2**]**;**

**Print**[**"Estimated error is :", Abs**[**x2**-**x1**]]**;**

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1 x1=2

Nmax=20

epsilon= 1

1 000 000

f[x]:=Cos[x]

In1th number of iterations the root is :1.5649 estimated error is: 0.435096

In2th number of iterations the root is :1.57098 estimated error is: 0.0060742

In3th number of iterations the root is :1.5708 estimated error is: 0.000182249 Return[1.5708]

root is : 1.5708

Estimated error is :1.02185×10 -9

-

|  |  |
| --- | --- |
| 1.0  0.5 |  |
| 1  -0.5  -1.0 | 1 2 3 |

**x0**=**Input**[**"Enter first guess: "**]**; x1**=**Input**[**"Enter scond guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"x1**=**", x1**]**; Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**x ^ 3**-**5**\***x**+**1; Print**[**"f**[**x**]**:**=**", f**[**x**]] **For****i**=**1, i**≤**Nmax, i**++**,**

**x2**=**N****x1**-**f**[**x**] /**. x**→**x1**\***x1**-**x0**  **f**[**x**] /**. x**→**x1**-**f**[**x**] /**. x**→**x0****; If**[**Abs**[**x1**-**x2**] <**eps, Return**[**x2**]**, x0**=**x1; x1**=**x2**]**;**

**Print**[**"In", i, "th number of iterations the root is :", x2**]**;**

**Print**[**"estimated error is: ", Abs**[**x1**-**x0**]]**; Print**[**"root is : ", x2**]**;**

**Print**[**"Estimated error is :", Abs**[**x2**-**x1**]]**;**

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1 x1=2

Nmax=20

epsilon= 1

1 000 000

f[x]:=1-5 x+x 3

In1th number of iterations the root is :2.5 estimated error is: 0.5

In2th number of iterations the root is :2.09756 estimated error is: 0.402439

In3th number of iterations the root is :2.12134 estimated error is: 0.0237786

In4th number of iterations the root is :2.12859 estimated error is: 0.0072456

In5th number of iterations the root is :2.12842 estimated error is: 0.000166952 Return[2.12842]

root is : 2.12842

Estimated error is :8.77361×10 -7

|  |  |
| --- | --- |
| 10  5 |  |
| 1 | 1 2 3 |

In[1]:= **x0**=**Input**[**"Enter first guess: "**]**; x1**=**Input**[**"Enter scond guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

-

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"x1**=**", x1**]**; Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**Cos**[**x**] -**x**\***Exp**[**x**]**;**

**Print**[**"f**[**x**]**:**=**", f**[**x**]] **For****i**=**1, i**≤**Nmax, i**++**,**

**x2**=**N****x1**-**f**[**x**] /**. x**→**x1**\***x1**-**x0**  **f**[**x**] /**. x**→**x1**-**f**[**x**] /**. x**→**x0****; If**[**Abs**[**x1**-**x2**] <**eps, Return**[**x2**]**, x0**=**x1; x1**=**x2**]**;**

**Print**[**"In", i, "th number of iterations the root is :", x2**]**;**

**Print**[**"estimated error is: ", Abs**[**x1**-**x0**]]**; Print**[**"root is : ", x2**]**;**

**Print**[**"Estimated error is :", Abs**[**x2**-**x1**]]**;**

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1 x1=2

Nmax=20 epsilon=1.×10 -6 f[x]:=-ⅇx x+Cos[x]

In1th number of iterations the root is :0.832673 estimated error is: 1.16733

In2th number of iterations the root is :0.728779 estimated error is: 0.103894

In3th number of iterations the root is :0.562401 estimated error is: 0.166377

In4th number of iterations the root is :0.524782 estimated error is: 0.0376189

In5th number of iterations the root is :0.518014 estimated error is: 0.00676874

In6th number of iterations the root is :0.517759 estimated error is: 0.0002547

In7th number of iterations the root is :0.517757 estimated error is: 1.50138×10 -6

Out[11]= Return[0.517757]

root is : 0.517757

Estimated error is :3.22103×10 -10

-1

1

2

3

-10

-20

-30

-40

-50

-60

Out[14]=

Practical 2(b) Prakhar Khugshal || BSc(Hons) Computer Science || Semester IV || 20211441

Regular Falsi

Q1

**x0**=**Input**[**"Enter first guess: "**]**; x1**=**Input**[**"Enter scond guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"x1**=**", x1**]**; Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**Cos**[**x**]**;**

**Print**[**"f**(**x**)**:**=**", f**[**x**]]**; If****N**[**f**[**x0**] \***f**[**x1**]] >**0,**

**Print**[**"These values does not satisfy the IVP so change the values "**]**, For****i**=**1, i**≤**Nmax, i**++**, a**=**N****x1**-**f**[**x1**] \***x1**-**x0**  **f**[**x1**] -**f**[**x0**]**, 16****;**

**If****Abs****x1**-**x0****2**<**eps, Return**[**N**[**a, 16**]]**, Print**[**i, "the iteration value is:", N**[**a16**]]**; Print**[**"Estimated error is: ", N**[**x1**-**x0, 16**]]**;**

**If**[**f**[**a**] \***f**[**x1**] >**0, x1**=**a, x0**=**a**]**;**

**Print**[**"Root is: ", N**[**a, 16**]]**; Print**[**"Estimated eror is:", N**[**x1**-**x0, 16**]]**;**

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1 x1=2

Nmax=20

epsilon= 1

1 000 000

f(x):=Cos[x]

1the iteration value is:a16

Estimated error is: 1.000000000000000

2the iteration value is:a16

Estimated error is: 0.435095624108422 3the iteration value is:a16

Estimated error is: 0.006074198643440 4the iteration value is:a16

Estimated error is: 0.000182248761967 5the iteration value is:a16

Estimated error is: 0.00018224774012 6the iteration value is:a16 Estimated error is: 0.00018224774012 7the iteration value is:a16 Estimated error is: 0.0001822477401 8the iteration value is:a16 Estimated error is: 0.0001822477401 9the iteration value is:a16 Estimated error is: 0.0001822477401 10the iteration value is:a16 Estimated error is: 0.000182247740 11the iteration value is:a16 Estimated error is: 0.000182247740 12the iteration value is:a16 Estimated error is: 0.000182247740 13the iteration value is:a16 Estimated error is: 0.000182247740 14the iteration value is:a16 Estimated error is: 0.00018224774 15the iteration value is:a16 Estimated error is: 0.00018224774 16the iteration value is:a16 Estimated error is: 0.00018224774 17the iteration value is:a16 Estimated error is: 0.0001822477 18the iteration value is:a16 Estimated error is: 0.0001822477 19the iteration value is:a16 Estimated error is: 0.0001822477

20the iteration value is:a16 Estimated error is: 0.000182248 Root is: 1.570796327

Estimated eror is:0.000182248

-

|  |  |
| --- | --- |
| 1.0  0.5 |  |
| 1  -0.5  -1.0 | 1 2 3 |

Q2

**x0**=**Input**[**"Enter first guess: "**]**; x1**=**Input**[**"Enter scond guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"x1**=**", x1**]**; Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**x ^ 3**-**5**\***x**+**1;**

**Print**[**"f**(**x**)**:**=**", f**[**x**]]**; If****N**[**f**[**x0**] \***f**[**x1**]] >**0,**

**Print**[**"These values does not satisfy the IVP so change the values "**]**, For****i**=**1, i**≤**Nmax, i**++**, a**=**N****x1**-**f**[**x1**] \***x1**-**x0**  **f**[**x1**] -**f**[**x0**]**, 16****;**

**If****Abs****x1**-**x0****2**<**eps, Return**[**N**[**a, 16**]]**, Print**[**i, "the iteration value is:", N**[**a16**]]**; Print**[**"Estimated error is: ", N**[**x1**-**x0, 16**]]**;**

**If**[**f**[**a**] \***f**[**x1**] >**0, x1**=**a, x0**=**a**]**;**

**Print**[**"Root is: ", N**[**a, 16**]]**; Print**[**"Estimated eror is:", N**[**x1**-**x0, 16**]]**;**

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1 x1=2

Nmax=20

epsilon= 1

1 000 000

f(x):=1-5 x+x 3

These values does not satisfy the IVP so change the values

-

|  |  |
| --- | --- |
| 10  5 |  |
| 1 | 1 2 3 |

Q3

**x0**=**Input**[**"Enter first guess: "**]**; x1**=**Input**[**"Enter scond guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"x1**=**", x1**]**; Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**Cos**[**x**] -**x**\***e^ x;**

**Print**[**"f**(**x**)**:**=**", f**[**x**]]**; If****N**[**f**[**x0**] \***f**[**x1**]] >**0,**

**Print**[**"These values does not satisfy the IVP so change the values "**]**, For****i**=**1, i**≤**Nmax, i**++**, a**=**N****x1**-**f**[**x1**] \***x1**-**x0**  **f**[**x1**] -**f**[**x0**]**, 16****;**

**If****Abs****x1**-**x0****2**<**eps, Return**[**N**[**a, 16**]]**, Print**[**i, "the iteration value is:", N**[**a16**]]**; Print**[**"Estimated error is: ", N**[**x1**-**x0, 16**]]**;**

**If**[**f**[**a**] \***f**[**x1**] >**0, x1**=**a, x0**=**a**]**;**

**Print**[**"Root is: ", N**[**a, 16**]]**; Print**[**"Estimated eror is:", N**[**x1**-**x0, 16**]]**;**

**Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1 x1=2

Nmax=20

epsilon= 1

1 000 000

f(x):=-ex x+Cos[x]

|  |  |
| --- | --- |
| 1.0  0.5 |  |
| 1  -0.5  -1.0 | 1 2 3 |

-

Practical 3 Prakhar Khugshal | BSC(hons) CS | 20211441 | IV semester

Newton Raphson Method

Q1

**x0**=**Input**[**"Enter first guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**Cos**[**x**]**; Print**[**"f**[**x**]**:**=**", f**[**x**]] **Print**[**"f'**[**x**]**:**=**", D**[**f**[**x**]**, x**]]**;**

**For****i**=**1, i**≤**Nmax, i**++**, x1**=**N****x0**-**f**[**x**] /**. x**→**x0**  **D**[**f**[**x**]**, x**] /**. x**→**x0****; If**[**Abs**[**x1**-**x0**] <**eps, Return**[**x1**]**, x0p**=**x0; x0**=**x1**]**;**

**Print**[**"In", i, "Th number of iteration the root is :", x1**]**;**

**Print**[**"estimated error is:", Abs**[**x1**-**x0p**]]**; Print**[**"The final approximationof the root is :", x1**]**; Print**[**"estimated error is :", Abs**[**x1**-**x0**]]**; Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1

Nmax=2 epsilon=10 f[x]:=Cos[x]

f'[x]:=-Sin[x]

Return[1.64209]

The final approximationof the root is :1.64209 estimated error is :0.642093

-

|  |  |
| --- | --- |
| 1.0  0.5 |  |
| 1  -0.5  -1.0 | 1 2 3 |

Q2

**x0**=**Input**[**"Enter first guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**x ^ 3**-**5**\***x**+**1; Print**[**"f**[**x**]**:**=**", f**[**x**]] **Print**[**"f'**[**x**]**:**=**", D**[**f**[**x**]**, x**]]**;**

**For****i**=**1, i**≤**Nmax, i**++**, x1**=**N****x0**-**f**[**x**] /**. x**→**x0**  **D**[**f**[**x**]**, x**] /**. x**→**x0****; If**[**Abs**[**x1**-**x0**] <**eps, Return**[**x1**]**, x0p**=**x0; x0**=**x1**]**;**

**Print**[**"In", i, "Th number of iteration the root is :", x1**]**;**

**Print**[**"estimated error is:", Abs**[**x1**-**x0p**]]**; Print**[**"The final approximationof the root is :", x1**]**; Print**[**"estimated error is :", Abs**[**x1**-**x0**]]**; Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1

Nmax=20

epsilon= 1

1 000 000

f[x]:=1-5 x+x 3

f'[x]:=-5+3 x 2

In1Th number of iteration the root is :-0.5 estimated error is:1.5

In2Th number of iteration the root is :0.294118 estimated error is:0.794118

In3Th number of iteration the root is :0.200215 estimated error is:0.093903

In4Th number of iteration the root is :0.201639 estimated error is:0.00142474

Return[0.20164]

The final approximationof the root is :0.20164 estimated error is :2.50538×10 -7

-

|  |  |
| --- | --- |
| 10  5 |  |
| 1 | 1 2 3 |

Q3

**x0**=**Input**[**"Enter first guess: "**]**; Nmax**=**Input**[**"Enter maximum of iterations : "**]**;**

**eps**=**Input**[**"Enter the value of covergence parameter: "**]**; Print**[**"x0**=**", x0**]**;**

**Print**[**"Nmax**=**", Nmax**]**; Print**[**"epsilon**=**", eps**]**; f**[**x**\_]**:**=**Cos**[**x**] -**x**\***Exp**[**x**]**;**

**Print**[**"f**[**x**]**:**=**", f**[**x**]] **Print**[**"f'**[**x**]**:**=**", D**[**f**[**x**]**, x**]]**;**

**For****i**=**1, i**≤**Nmax, i**++**, x1**=**N****x0**-**f**[**x**] /**. x**→**x0**  **D**[**f**[**x**]**, x**] /**. x**→**x0****; If**[**Abs**[**x1**-**x0**] <**eps, Return**[**x1**]**, x0p**=**x0; x0**=**x1**]**;**

**Print**[**"In", i, "Th number of iteration the root is :", x1**]**;**

**Print**[**"estimated error is:", Abs**[**x1**-**x0p**]]**; Print**[**"The final approximationof the root is :", x1**]**; Print**[**"estimated error is :", Abs**[**x1**-**x0**]]**; Plot**[**f**[**x**]**,**{**x,**-**1, 3**}]

x0=1

Nmax=20 epsilon=1.×10 -6 f[x]:=-ⅇx x+Cos[x]

f'[x]:=-ⅇx -ⅇ x x-Sin[x]

In1Th number of iteration the root is :0.653079 estimated error is:0.346921

In2Th number of iteration the root is :0.531343 estimated error is:0.121736

In3Th number of iteration the root is :0.51791 estimated error is:0.0134335

In4Th number of iteration the root is :0.517757 estimated error is:0.00015253 Return[0.517757]

The final approximationof the root is :0.517757 estimated error is :1.94824×10 -8

-1

1

2

3

-10

-20

-30

-40

-50

-60

Practical 4

Prakhar Khugshal | BSc(H) Computer Science | Sem - IV | 20211441

1. Gaussian Elimination Method

Q1. Solve the following system of equations by using Gaussian Elimination Method

2x1-3x2+10x3=-2 x1-2x2+3x3=-2

-x1+3x2+x3=4

**MatrixForm**[**A**= {{**2,**-**3, 10,**-**2**}**,**{**1,**-**2, 3,**-**2**}**,**{-**1, 3, 1, 4**}}]

2-3 10-2

1-2 3-2

-1 3 1 4

**MatrixForm**[**A**= {**A**[[**2**]]**, A**[[**1**]]**, A**[[**3**]]}]

1-2 3-2

2-3 10-2

-1 3 1 4

**MatrixForm**[**A**= {**A**[[**1**]]**, A**[[**2**]] -**2 A**[[**1**]]**, A**[[**3**]] +**A**[[**1**]]}]

1-2 3-2

0 1 4 2

0 1 4 2

**MatrixForm**[**A**= {**A**[[**1**]]**, A**[[**2**]]**, A**[[**3**]] -**A**[[**2**]]}]

1-2 3-2

0 1 4 2

0 0 0 0

**2** *Prakhar 20211441.nb*

**Solve**[{**x1**-**2 x2**+**3 x3**⩵-**2, x2**+**4 x3**⩵**2**}**,**{**x3, x2, x1**}]

Solve : Equations may not give solutions for all "solve" variables.

{{x2→2-4 x3, x1→2-11 x3}}

Q1. Solve the following system of equations by using Gaussian Elimination Method

2x1+x2+x3=10

3x1+2x2+3x3=18 x1+4x2+9x3=16

**MatrixForm**[**A**= {{**2, 1, 1, 10**}**,**{**3, 2, 3, 18**}**,**{**1, 4, 9, 16**}}]

2 1 1 10

3 2 3 18

1 4 9 16

**MatrixForm****A**=**A**[[**1**]]**, A**[[**2**]] -**3****2 A**[[**1**]]**, A**[[**3**]] -**1****2 A**[[**1**]]

2 1 1 10

0 1 3

3

2 2

0 7 17

11

2 2

**MatrixForm**[**A**= {**A**[[**1**]]**, A**[[**2**]]**, A**[[**3**]] -**7 A**[[**2**]]}]

2 1 1 10

0 1 3

3

2 2

0 0-2-10

**Solve****2 x1**+**x2**+**x3**⩵**10, 1**  **2 x2**+**3**  **2 x3**⩵**3,**-**2 x3**⩵-**10****,**{**x3, x2, x1**}

{{x3→5, x2→-9, x1→7}}

1. Gauss Jordan Elimination Method

Q1. Solve the following system of equations by using Gauss Jordan Elimination Method

2x1+x2+x3=10

*Prakhar 20211441.nb* **3**

3x1+2x2+3x3=18 x1+4x2+9x3=16

**MatrixForm**[**B**= {{**2, 1, 1, 10**}**,**{**3, 2, 3, 18**}**,**{**1, 4, 9, 16**}}]

2 1 1 10

3 2 3 18

1 4 9 16

**MatrixForm**[**RowReduce**[**B**]]

|  |  |
| --- | --- |
| 1 | 0 0 7 |
| 0 | 1 0-9 |
| 0 | 0 1 5 |

**Solve**[{**x1**⩵**7, x2**⩵-**9, x3**⩵**5**}**,**{**x3, x2, x1**}]

{{x3→5, x2→-9, x1→7}}

Inverse

**MatrixForm**[**B**= {{**2, 1, 1, 1, 0, 0**}**,**{**3, 2, 3, 0, 1, 0**}**,**{**1, 4, 9, 0, 0, 1**}}]

2 1 1 1 0 0

3 2 3 0 1 0

1 4 9 0 0 1

**MatrixForm**[**RowReduce**[**B**]]

5 - 1

|  |  |
| --- | --- |
| 1 | 0 0-3 |
| 0 | 1 0 12- |
| 0 | 0 1-5 |

2 2

17 3

2 2

7 - 1

2 2

Practical 5(a)

Prakhar Khugshal| BSc(H) Computer Science | Sem - IV | 20211441

Gauss Jacobi method

Question 1 :

**GaussJacobi**[**A0**\_**, b0**\_**, X0**\_**, maxiter**\_]**:**=

**Module**{**A**=**N**[**A0**]**, b**=**N**[**b0**]**, xk**=**X0, xk1, i, j, k**=**0, n, m, OutputDetails**}**, size**=**Dimensions**[**A**]**;**

**n**=**size**[[**1**]]**;**

**m**=**size**[[**2**]]**; If**[**n**≠**m,**

**Print**[**"Not a square matrix, cannot proceed with Gauss Jacobi method"**]**; Return**[]]**;**

**OutputDetails**= {**xk**}**; xk1**=**Table**[**0,**{**n**}]**; While****k**<**maxiter, For****i**=**1, i**≤**n, i**++**,**

**1 i**-**1 n**

**xk1**[[**i**]] =

**b**[[**i**]] -  **A**[[**i, j**]] \***xk**[[**j**]] - 

**A**[[**i, j**]] \***xk**[[**j**]]

**;****;**

**k**++**;**

**A**[[**i, i**]]

**j**=**1**

**j**=**i**+**1**

**OutputDetails**=**Append**[**OutputDetails, xk1**]**; xk**=**xk1;****;**

**colHeading**=**Table**[**X**[**s**]**,**{**s, 1, n**}]**; Print**[**NumberForm**[**TableForm**[**OutputDetails,**

**TableHeadings**→{**None, colHeading**}]**, 6**]]**; Print**[**"No. of iterations performed ", maxiter**]**;****;**

**A**= {{**5, 1, 2**}**,**{-**3, 9, 4**}**,**{**1, 2,**-**7**}}**;**

**b**= {**10,**-**14,**-**33**}**;**

**X0**= {**0, 0, 0**}**;**

**GaussJacobi**[**A, b, X0, 15**]

**2** *Prakhar 5(a).nb*

X[1]X[2]X[3]

0 0 0

2.-1.55556 4.71429

|  |  |
| --- | --- |
| 0.425397-2.98413 | 4.55556 |
| 0.774603-3.43845 | 3.92245 |
| 1.11871-3.04067 | 3.84253 |
| 1.07112-2.89044 | 4.00534 |
| 0.975953-2.97867 | 4.04146 |
| 0.979148-3.02644 | 4.00266 |
| 1.00422-3.00813 | 3.98947 |
| 1.00584-2.99391 | 3.99828 |
| 0.99947-2.99729 | 4.00257 |
| 0.998428-3.00132 | 4.0007 |
| 0.999985-3.00083 | 3.9994 |
| 1.00041-2.99974 | 3.99976 |
| 1.00004-2.99976 | 4.00013 |
| 0.999898-3.00004 | 4.00008 |

No. of iterations performed 15

Question 1I:

**GaussJacobi**[**A0**\_**, b0**\_**, X0**\_**, maxiter**\_]**:**=

**Module**{**A**=**N**[**A0**]**, b**=**N**[**b0**]**, xk**=**X0, xk1, i, j, k**=**0, n, m, OutputDetails**}**, size**=**Dimensions**[**A**]**;**

**n**=**size**[[**1**]]**;**

**m**=**size**[[**2**]]**; If**[**n**≠**m,**

**Print**[**"Not a square matrix, cannot proceed with Gauss Jacobi method"**]**; Return**[]]**;**

**OutputDetails**= {**xk**}**; xk1**=**Table**[**0,**{**n**}]**; While****k**<**maxiter, For****i**=**1, i**≤**n, i**++**,**

**1 i**-**1 n**

**xk1**[[**i**]] =

**b**[[**i**]] -  **A**[[**i, j**]] \***xk**[[**j**]] - 

**A**[[**i, j**]] \***xk**[[**j**]]

**;****;**

**k**++**;**

**A**[[**i, i**]]

**j**=**1**

**j**=**i**+**1**

**OutputDetails**=**Append**[**OutputDetails, xk1**]**; xk**=**xk1;****;**

**colHeading**=**Table**[**X**[**s**]**,**{**s, 1, n**}]**; Print**[**NumberForm**[**TableForm**[**OutputDetails,**

**TableHeadings**→{**None, colHeading**}]**, 6**]]**; Print**[**"No. of iterations performed ", maxiter**]**;****;**

**A**= {{**5, 1, 2**}**,**{-**3, 9, 4**}**,**{**1, 2,**-**7**}**,**{**2, 1, 3**}}**;**

**b**= {**10,**-**14,**-**33**}**;**

**X0**= {**0, 0, 0**}**;**

**GaussJacobi**[**A, b, X0, 15**]

Not a square matrix, cannot proceed with Gauss Jacobi method

*Prakhar 5(a).nb* **3**

Question 1II :

**GaussJacobi**[**A0**\_**, b0**\_**, X0**\_**, maxiter**\_]**:**=

**Module**{**A**=**N**[**A0**]**, b**=**N**[**b0**]**, xk**=**X0, xk1, i, j, k**=**0, n, m, OutputDetails**}**, size**=**Dimensions**[**A**]**;**

**n**=**size**[[**1**]]**;**

**m**=**size**[[**2**]]**; If**[**n**≠**m,**

**Print**[**"Not a square matrix, cannot proceed with Gauss Jacobu method"**]**; Return**[]]**;**

**OutputDetails**= {**xk**}**; xk1**=**Table**[**0,**{**n**}]**; While****k**<**maxiter, For****i**=**1, i**≤**n, i**++**,**

**1 i**-**1 n**

**xk1**[[**i**]] =

**b**[[**i**]] -  **A**[[**i, j**]] \***xk**[[**j**]] - 

**A**[[**i, j**]] \***xk**[[**j**]]

**;****;**

**k**++**;**

**A**[[**i, i**]]

**j**=**1**

**j**=**i**+**1**

**OutputDetails**=**Append**[**OutputDetails, xk1**]**; xk**=**xk1;****;**

**colHeading**=**Table**[**X**[**s**]**,**{**s, 1, n**}]**; Print**[**NumberForm**[**TableForm**[**OutputDetails,**

**TableHeadings**→{**None, colHeading**}]**, 6**]]**; Print**[**"No. of iterations performed ", maxiter**]**;****;**

**A**= {{**5, 1, 2**}**,**{-**3, 9, 4**}**,**{**1, 9,**-**7**}}**;**

**b**= {**11,**-**14,**-**30**}**;**

**X0**= {**0, 0, 0**}**;**

**GaussJacobi**[**A, b, X0, 15**]

X[1]X[2]X[3]

0 0 0

2.2-1.55556 4.28571

|  |  |
| --- | --- |
| 0.796825-2.72698 | 2.6 |
| 1.7054-2.4455 | 0.893424 |
| 2.33173-1.38417 | 1.38512 |
| 1.92278-1.39392 | 2.83918 |
| 1.34311-2.17648 | 2.76821 |
| 1.52801-2.33817 | 1.67925 |
| 1.99593-1.79255 | 1.49779 |
| 1.9594-1.55593 | 2.26614 |
| 1.60473-1.9096 | 2.56515 |
| 1.55586-2.16071 | 2.05977 |
| 1.80824-1.95239 | 1.72992 |
| 1.89851-1.72166 | 2.03382 |
| 1.7308-1.82664 | 2.34336 |
| 1.62798-2.02011 | 2.18444 |
| No. of iterations | performed 15 |

Practical 6

Prakhar Khugshal | B.Sc (Hons) Computer Science | IV Semester | 20211441

Lagrange Interpolation Polynomial

In[44]:= **LagrangePolynomial**[**x0**\_**, f0**\_]**:**= **Module**{**xi**=**x0, fi**=**f0, n, m, polynomial**}**,**

**n**=**Length**[**xi**]**; m**=**Length**[**fi**]**; If**[**n**≠**m,**

**Print**[**"List of points and function values are not of same size"**]**; Return**[]**;**]**;**

**For****i**=**1, i**≤**n, i**++**,**

**L**[**i, x**\_] =

**i**-**1 x**-**xi**[[**j**]]



**j**=**1 xi**[[**i**]] -**xi**[[**j**]]

**n**

**n**  **x**-**xi**[[**j**]]



**j**=**i**+**1 xi**[[**i**]] -**xi**[[**j**]]

**;****;**

**polynomial**[**x**\_] =  **L**[**k, x**] \***fi**[[**k**]]**;**

**k**=**1**

**Return**[**polynomial**[**x**]]**;**

Q1

In[45]:= **nodes**= {**0, 1, 3**}**;**

**values**= {**1, 3, 55**}**;**

**LagrangePolynomial**[**x**\_] =**LagrangePolynomial**[**nodes, values**] **Expand** **1** **1**-**x** **3**-**x**+ **3** **3**-**x****x**+ **55** -**1**+**x****x**

**3 2 6**

Out[47]= 1 1-x 3-x+ 3 3-xx+ 55 -1+xx

3 2 6

Out[48]= 1-6 x+8 x 2

**2** *Prakhar practical 6.nb*

In[49]:= **nodes**= {**0, 1, 3**}**;**

**values**= {**1, 3**}**;**

**LagrangePolynomial**[**x**\_] =**LagrangePolynomial**[**nodes, values**]

List of points and function values are not of same size

P2

In[52]:= **nodes**= {**1, 3, 5, 7, 9**}**;**

**values**= {**N**[**Log**[**1**]]**, N**[**Log**[**3**]]**, N**[**Log**[**5**]]**, N**[**Log**[**7**]]**, N**[**Log**[**9**]]}**;**

**LagrangePolynomial**[**x**\_] =**LagrangePolynomial**[**nodes, values**]

Out[54]= 0.+0.0114439(5-x) (7-x)9-x -1+x+0.0251475(7-x)9-x -3+x -1+x+

0.02026999-x(-5+x)-3+x -1+x+0.00572194(-7+x) (-5+x)-3+x -1+x

In[55]:= **Simplify****0.`**+**0.011443878006959476`**(**5**-**x**) (**7**-**x**)**9**-**x** -**1**+**x**+

**0.025147467381782817`**(**7**-**x**)**9**-**x** -**3**+**x** -**1**+**x**+

**0.020269897385992844`****9**-**x**(-**5**+**x**)-**3**+**x** -**1**+**x**+

**0.005721939003479738`**(-**7**+**x**) (-**5**+**x**)-**3**+**x** -**1**+**x**

Out[55]= -0.987583+1.18991 x-0.223608 x 2 +0.0221231 x 3 -0.000844369 x 4

In[56]:= **Plot**[{**LagrangePolynomial**[**x**]**, Log**[**x**]}**,**{**x, 1, 10**}**,**

**Ticks**→{**Range**[**0, 10**]}**, PlotLegends**→**"Expressions"**]

2

1

2

3

4

5

6

7

8

9

10

Out[56]=

LagrangePolynomial(x) log(x)

In[57]:= **nodes**= {-**1, 0, 1, 2**}**;**

**values**= {**5, 1, 1, 11**}**;**

**LagrangePolynomial**[**x**\_] =**LagrangePolynomial**[**nodes, values**]

Out[59]= - 5 1-x 2-xx+ 1 1-x 2-x 1+x+ 1 2-xx1+x+ 11 -1+xx1+x

6 2 2 6

In[60]:= **Simplify**- **5** **1**-**x** **2**-**x****x**+ **1** **1**-**x** **2**-**x** **1**+**x**+ **1** **2**-**x****x****1**+**x**+ **11** -**1**+**x****x****1**+**x**

**6 2 2 6**

Out[60]= 1-3 x+2 x 2 +x 3

In[61]:= **LagrangePolynomial**[**1.5**]

Out[61]= 4.375

Practica 6(b)

Prakhar Khugshal | B.Sc(Hons) Computer Science | IV Semester | 20211441 |

Newton Divided Difference Interpolating polyomial

In[4]:= **NthDividedDiff**[**x0**\_**, f0**\_**, startindex**\_**, endindex**\_]**:**= **Module**{**x**=**x0, f**=**f0, i**=**startindex, j**=**endindex, answer**}**, If****i**⩵**j, Return**[**f**[[**i**]]]**,**

**answer**=

**NthDividedDiff**[**x, f, i**+**1, j**] -**NthDividedDiff**[**x, f, i, j**-**1**]  **x**[[**j**]] -**x**[[**i**]]**; Return**[**answer**]**;**

**;**

**x**= {**0, 1, 3**}**;**

**f**= {**1, 3, 55**}**;**

**NthDividedDiff**[**x, f, 2, 3**]

Out[7]= 26

In[8]:= **NthDividedDiff**[**x, f, 1, 3**]

Out[8]= 8

In[9]:= **x**= {-**1, 0, 1, 2**}**;**

**f**= {**5, 1, 1, 11**}**;**

**NthDividedDiff**[**x, f, 1, 2**]

Out[11]= -4

In[12]:= **NthDividedDiff**[**x, f, 2, 3**]

Out[12]= 0

In[13]:= **NthDividedDiff**[**x, f, 1, 3**]

Out[13]= 2

In[14]:= **NthDividedDiff**[**x, f, 2, 4**]

Out[14]= 5

**2** *Prakhar Practical 6(b).nb*

In[15]:= **NthDividedDiff**[**x, f, 1, 4**]

Out[15]= 1

Q2

In[21]:= **NewtonDDPoly**[**x0**\_**, f0**\_]**:**=

**Module**{**x1**=**x0, f**=**f0, n, newtonPolynomial, k, j**}**, n**=**Length**[**x1**]**;**

**newtonPolynomial**[**Y**\_] =**0;**

**For****i**=**1, i**≤**n, i**++**, prod**[**Y**\_] =**1; For****k**=**1, k**≤**i**-**1, k**++**,**

**prod**[**Y**\_] =**prod**[**Y**] \***y**-**x1**[[**k**]]**;**

**newtonPolynomial**[**Y**\_] =**newtonPolynomial**[**Y**] +**NthDividedDiff**[**x1, f, 1, i**] \***prod**[**Y**]**; Return**[**newtonPolynomial**[**Y**]]**;****;**

**nodes**= {**0, 1, 3**}**;**

**values**= {**1, 3, 55**}**; NewtonDDPoly**[**nodes, values**]

Out[24]= 1+2 y+8-1+yy

In[25]:= **Simplify****1**+**2 y**+**8**-**1**+**y****y**

Out[25]= 1-6 y+8 y 2

Practical 7 (a)

Prakhar Khugshal | BSc(H) Computer Science | Sem - IV | 20211441

Trapezoidal Method

Q1.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**=**Log**[**x**]**; sumodd**=**0; sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Tn**=**h****2**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****; Print**[**"For n**=**", n, " Trapezoidal estimate is :", Tn**] **in**=**Integrate**[**Log**[**x**]**,**{**x, 4, 5.2**}]

**Print**[**"True value is ", in**] **Print**[**"Absolute error is ", Abs**[**Tn**-**in**]]

For n=6 Trapezoidal estimate is :26.8772

1.82785

True value is 1.82785 Absolute error is 25.0494

**2** *Prakhar Practical 7(a) (20211435).nb*

Q2.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**=**Sin**[**x**]**; sumodd**=**0; sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Tn**=**h****2**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****;**

**Print**[**"For n**=**", n, " Trapezoidal estimate is :", Tn**]

π

**in1**=**Integrate****Sin**[**x**]**,****x, 0,** 

**2**

**Print**[**"True value is ", in1**]

**Print**[**"Absolute error is ", Abs**[**Tn**-**in1**]] For n=6 Trapezoidal estimate is :-0.944145 1

True value is 1

Absolute error is 1.94415

Q3.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**=**Sin**[**x**] -**Log**[**x**] +**Exp**[**x**]**; sumodd**=**0;**

**sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Tn**=**h****2**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****; Print**[**"For n**=**", n, " Trapezoidal estimate is :", Tn**] **in1**=**Integrate**[**Sin**[**x**] -**Log**[**x**] +**Exp**[**x**]**,**{**x, 0.2, 1.4**}]

**Print**[**"True value is ", in1**] **Print**[**"Absolute error is ", Abs**[**Tn**-**in1**]]

For n=6 Trapezoidal estimate is :5.92567×10 8

4.05095

True value is 4.05095 Absolute error is 5.92567×10 8

*Prakhar Practical 7(a) (20211435).nb* **3**

Q4.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**= **1 ;**

**1**+**x^ 2**

**sumodd**=**0; sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Tn**=**h****2**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****; Print**[**"For n**=**", n, " Trapezoidal estimate is :", Tn**]

**in1**=**Integrate** **1 ,**{**x, 0, 1**}

**1**+**x^ 2**

**Print**[**"True value is ", in1**] **Print**[**"Absolute error is ", Abs**[**Tn**-**in1**]]

For n=6 Trapezoidal estimate is :0.0501042

π

4

True value is π

4

Absolute error is 0.735294

Practical 7 (b)

N. Prakhar Khugshal | BSc(H) Computer Science | Sem - IV | 20211441

Simpson Method

Q1.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**= **1 ;**

**x**

**sumodd**=**0; sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Sn**=**h****3**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****; Print**[**"For n**=**", n, " Simpson estimate is :", Sn**]

**in1**=**Integrate** **1 ,**{**x, 1, 2**}

**x**

**Print**[**"True value is ", in1**] **Print**[**"Absolute error is ", Abs**[**Sn**-**in1**]]

For n=6 Simpson estimate is :0.463252

Log[2]

True value is Log[2] Absolute error is 0.229896

Q2.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**=**Log**[**x**]**; sumodd**=**0; sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Sn**=**h****3**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****; Print**[**"For n**=**", n, " Simpson estimate is :", Sn**] **in1**=**Integrate**[**Log**[**x**]**,**{**x, 4, 5.2**}]

**Print**[**"True value is ", in1**] **Print**[**"Absolute error is ", Abs**[**Sn**-**in1**]]

For n=6 Simpson estimate is :17.9182

1.82785

True value is 1.82785 Absolute error is 16.0903

Q3.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**=**Sin**[**x**] -**Log**[**x**] +**Exp**[**x**]**; sumodd**=**0;**

**sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Sn**=**h****3**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****; Print**[**"For n**=**", n, " Simpson estimate is :", Sn**] **in1**=**Integrate**[**Sin**[**x**] -**Log**[**x**] +**Exp**[**x**]**,**{**x, 0.2, 1.4**}]

**Print**[**"True value is ", in1**] **Print**[**"Absolute error is ", Abs**[**Sn**-**in1**]]

For n=6 Simpson estimate is :3.95045×10 8

4.05095

True value is 4.05095 Absolute error is 3.95045×10 8

*Prakhar Practical 7(b) (20211441).nb* **3**

Q4.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**=**Sin**[**x**]**; sumodd**=**0; sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Sn**=**h****3**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****;**

**Print**[**"For n**=**", n, " Simpson estimate is :", Sn**]

π

**in1**=**Integrate****Sin**[**x**]**,****x, 0,** 

**2**

**Print**[**"True value is ", in1**]

**Print**[**"Absolute error is ", Abs**[**Sn**-**in1**]]

For n=6 Simpson estimate is :-0.62943

1

True value is 1

Absolute error is 1.62943

Q5.

**a**=**Input**[**"Enter the left end point: "**]**; b**=**Input**[**"Enter the right end point: "**]**;**

**n**=**Input**[**"Enter the number of sub intervals to be formed: "**]**; h**=**b**-**a** **n;**

**y**=**Table**[**a**+**i**\***h,**{**i, 1, n**}]**;**

**f**[**x**]**:**=**x^ 0.5**\***Exp**[**x**]**; sumodd**=**0;**

**sumeven**=**0;**

**For**[**i**=**1, i**<**n, i**+=**2, sumodd**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**; For**[**i**=**2, i**<**n, i**+=**2, sumeven**+=**2**\***f**[**x**] /**. x**→**y**[[**i**]]]**;**

**Sn**=**h****3**\***f**[**x**] /**. x**→**a**+**N**[**sumodd**] +**N**[**sumeven**] +**f**[**x**] /**. x**→**b****; Print**[**"For n**=**", n, " Simpson estimate is :", Sn**] **in1**=**Integrate****x ^ 0.5**\***Exp**[**x**]**,**{**x, 1, 2**}

**Print**[**"True value is ", in1**] **Print**[**"Absolute error is ", Abs**[**Sn**-**in1**]]

For n=6 Simpson estimate is :1.73692×10 9

5.85023

True value is 5.85023 Absolute error is 1.73692×10 9