

B.Sc Physics (Major)
Semester I
Physics Core Course 1

Mathematical Physics
Laboratory Notebook

Name : Dhiman Sarkar

Roll Number : 1810093104502

Registration Number : 0091805030584

College Roll Number : 2180051

College : Sukanta Mahavidyalaya

Session : 2018-2019

INDEX

	<u>Title</u>	<u>Page No.</u>
i.	Finding the Sum and Average of the all Even or Odd numbers from the first given n Natural Numbers	1
ii.	Find the greatest number and its position from a given set of numbers	2
iii.	Solving Polynomial Equations using Bisection Method Given Equation $x^3 - x^2 - 2 = 0$	3
iv.	Solving Polynomial Equations using Newton-Rhapson Method Given Equation $x^3 - x^2 - 2 = 0$	4
v.	Solving Polynomial Equations using Secant Method Given Equation $x^3 - x^2 - 2 = 0$	5
vi.	Numerical Integration using Trapezoidal Rule Given Function $f(x) = 1/(1+x^2)$	6
vii.	Numerical Integration using Trapezoidal Rule Given Function $f(x) = 1/(1+x^2)$	7
viii.	Finding the Velocity and Acceleration from a given set of data of Position and Time by using Forward Difference Numerical Differentiation	8
ix.	Finding the Velocity and Acceleration from a given set of data of Position and Time by using Forward Difference Numerical Differentiation	9
x.	Solving a ODE by using Euler's Method Given function $f'(x) = 3x^2 + 1$	10
xi.		
xii.		

```

1 // Finding the Sum and Average of the all Even or Odd numbers from the first given n
  Natural Numbers
2
3 #include<stdio.h>
4 #include<conio.h>
5 #include<string.h>
6
7 void main()
8 {
9     printf("Find the Sum of all the Even or Odd numbers from the first n Natural
  Numbers");
10
11     char choice[5];
12     int n, sum=0, avg, i, j=0; // n= Number of Natural Numbers, i=General purpose
  initilizer, j=Number Count
13
14     printf("\n\nPlease enter the last natural number : ");
15     scanf("%d",&n);
16
17     start: //For restart purpose
18
19     printf("\nWhich type of numbers (Even/Odd) do you want to calculate the sum from
  first %d Natural numbers? Answer (even/odd) : ",n);
20     scanf("%s",choice);
21
22     if(strcmp(choice, "even") == 0) i=2;
23     else if(strcmp(choice, "odd") == 0) i=1;
24     else {printf("Error!"); goto start;}
25
26     for(i; i<=n; i=i+2)
27     {
28         j++;
29         sum=sum+i;
30         avg=sum/j;
31     }
32
33     printf("\n\nSum of all the %s numbers from the first %d Natural Numbers is : %d",
  choice, n, sum);
34     printf("\n\nAverage of all the %s numbers (%d numbers) from the first %d Natural
  Numbers is : %d",choice,j,n,avg);
35
36     printf("\n\n");
37     getch();
38 }
39

```

Output :

1.

```
## Find the Sum of all the Even or Odd numbers from the first n Natural Numbers ##
```

```
Please enter the last natural number : 10
```

```
Which type of numbers (Even/Odd) do you want to calculate the sum from first 10  
Natural numbers? (even/odd)
```

```
Answer : even
```

```
Sum of all the even numbers from the first 10 Natural Numbers is : 30
```

```
Average of all the even numbers (5 numbers) from the first 10 Natural Numbers is : 6
```

2.

```
## Find the Sum of all the Even or Odd numbers from the first n Natural Numbers ##
```

```
Please enter the last natural number : 43
```

```
Which type of numbers (Even/Odd) do you want to calculate the sum from first 43  
Natural numbers? (even/odd)
```

```
Answer : odd
```

```
Sum of all the odd numbers from the first 43 Natural Numbers is : 484
```

```
Average of all the odd numbers (22 numbers) from the first 43 Natural Numbers is : 22
```

```

1 // Find the greatest number and its postition from a given set of numbers
2
3 #include<stdio.h>
4 #include<conio.h>
5
6 void main()
7 {
8     printf("## Find the greatest number and its postition from a given set of numbers
9     ##\n\n");
10
11     int n, pos, i;
12     printf("How many numbers are there?\n>>");
13     scanf("%d",&n);
14
15     float number[n], grtst_num;
16
17     for(i=0; i<n; i++)
18     {
19         printf("Enter the %th Number >>",i+1);
20         scanf("%f",&number[i]);
21     }
22
23     // Finding the Greatest number and its Position
24     grtst_num = number[0]; //Initilizing comparison with the first number
25     pos=1;
26     for(i=1; i<n; i++)
27     {
28         if(number[i]>=grtst_num)
29         {
30             grtst_num = number[i];
31             pos = i+1;
32         }
33         else ;
34     }
35
36     printf("\nThe Greatest number is :: %f and it's at the %dth Position", grtst_num,
37     pos);
38
39     getch();
40 }

```

Output :

```
## Find the greatest number and its postition from a given set of numbers ##
```

```
How many numbers are there?
```

```
>>11
```

```
Enter the 1th Number >>-54
```

```
Enter the 2th Number >>56
```

```
Enter the 3th Number >>12
```

```
Enter the 4th Number >>-9
```

```
Enter the 5th Number >>9
```

```
Enter the 6th Number >>15
```

```
Enter the 7th Number >>62
```

```
Enter the 8th Number >>-8
```

```
Enter the 9th Number >>0
```

```
Enter the 10th Number >>3
```

```
Enter the 11th Number >>4
```

```
The Greatest number is :: 62.000000 and it's at the 7th Position
```

```

1 // Solving Polynomial Equations using Bisection Method
2 // Given Function f(x) = x^3 -x^2 -2=0
3
4 #include<stdio.h>
5 #include<math.h>
6 #include<stdlib.h>
7 #include<conio.h>
8
9
10 // Function Prototype
11 float f(float x);
12 float tolerance(float x1, float x2);
13 int bisection_method (float x1, float x2, float TOL);
14
15 // main() Function
16 void main()
17 {
18     float TOL, x1, x2; //TOL = Desiered Tollerence X1,X2=Initial boundary
19     printf("##### This Program is to solve a equation by Bisection Method
20     #####\n\n");
21     printf("Please enter Tolerance : ");
22     scanf("%f",&TOL);
23
24     START: //For Restart purpose
25
26     printf("\nEnter the lower bound of the solution :: ");
27     scanf("%f", &x1);
28     printf("\nEnter the upper bound of the solution :: ");
29     scanf("%f", &x2);
30
31     if (bisection_method(x1,x2,TOL)==0) goto START;
32     else ;
33     getch();
34 }
35
36 // Defining function f(x)
37 float f(float x)
38 {
39     float fx = x*x*x -x*x +2; // Define f(x)
40     return fx;
41 }
42
43 //Defining Tolerance function
44 float tolerance(float x1, float x2)
45 {
46     float TOL = abs(x2-x1); // TOL> Tolerance = x2-x1
47     return TOL;
48 }
49
50
51 //Defining Bisection Method
52 int bisection_method (float x1, float x2, float TOL)
53 {
54     float x0;
55     int n; //n=number of itteration
56
57     if ( f(x1)*f(x2)>0)
58     {
59         printf ("\nSolution doesn't exists in the domain (%f,%f)", x1,x2);

```

```

60     return (0);
61
62 }
63 else ;
64
65     for(n=1; n>=1; n++)
66     {
67         x0 = (x1+x2)/2;
68
69         if (tolerance(x1,x2)<=TOL)
70         {
71             printf("\nSolution of the Polynomial equation is :: %f", x0);
72             printf("\nNumbner of Iteration :: %d\n\n", n);
73             break;
74         }
75         else if (f(x1)*f(x0)<=0) x2=x0;
76         else if (f(x2)*f(x0)<=0) x1=x0;
77         else {printf("Error!!");return 0;}
78     }
79     return 1;
80 }
81

```


Output :

```
##### This Program is to solve a equation by Bisection Method #####
```

```
Please enter Tolerance : .00001
```

```
Enter the lower bound of the solution :: 1
```

```
Enter the upper bound of the solution :: 5
```

```
Solution doesn't exists in the domain (1.000000,5.000000)
```

```
Enter the lower bound of the solution :: -8
```

```
Enter the upper bound of the solution :: 5
```

```
Solution of the Polynomial equation is :: -1.000001
```

```
Number of Iteration :: 22
```

```

1 // Solving Polynomial Equation by using Newton-Rhapson Method
2 // Given Function f(x) = x^3 -x^2 +2
3
4 #include<stdio.h>
5 #include<math.h>
6 #include<stdlib.h>
7 #include<conio.h>
8
9
10 // Function Prototype
11 float F(float x); // Original Function
12 float f(float x); // Differential Coefficient of F(x)
13 float tolerance(float x1, float x2);
14 int NR_method (float x1, float TOL);
15
16 // main() Function
17 void main()
18 {
19     float TOL, x1; //TOL = Desired Tolerance X1,X2=Initial boundary
20     printf("##### This Program is to solve a equation by Newton-Rhapson Method
21     #####\n\n");
22     printf("Please enter Tolerance : ");
23     scanf("%f",&TOL);
24
25     printf("\nEnter the initial approximation of the solution :: ");
26     scanf("%f", &x1);
27
28     NR_method(x1,TOL);
29
30 }
31
32
33 // Defining function f(x) | F(x) stands for anti-derivative i.e. the given function
34 float F(float x)
35 {
36     float Fx = x*x*x - x*x +2; // Define f(x)
37     return Fx;
38 }
39
40 // Defining Differential Co-efficient
41 float f(float x)
42 {
43     float fx = 3*x*x -4*x; // Define f'(x)
44     return fx;
45 }
46
47 //Defining Tolerance function
48 float tolerance(float x1, float x2)
49 {
50     float TOL = fabs(x2-x1);
51     return TOL;
52 }
53
54
55 //Defining Bisection Method
56 int NR_method (float x1, float TOL)
57 {
58     float x2;
59     int n; //n=number of iteration

```

```

60
61     for(n=1; n>0; n++)
62     {
63         x2 = x1 - F(x1)/f(x1); // p[n] = p[n-1]-f(p[n-1])/f'(p[n-1])
64
65         if (tolerance(x1,x2)<=TOL)
66         {
67             printf("\nSolution of the Polynomial equation is :: %f", x2);
68             printf("\nNumber of Iteration :: %d\n\n", n);
69             break;
70         }
71         else x1=x2;
72     }
73 }
74

```

Output :

This Program is to solve a equation by Newton-Rhapson Method

Please enter Tolerance : .00001

Enter the initial approximation of the solution :: 21

Solution of the Polynomial equation is :: -1.000001

Number of Iteration :: 23

```

1 // Solving Polynomial Equations using Secant Method
2 // Given Function  $f(x) = x^3 - x^2 - 2 = 0$ 
3
4 #include<stdio.h>
5 #include<math.h>
6 #include<stdlib.h>
7 #include<conio.h>
8
9 #define ERROR .0001
10
11
12 // Function Prototype
13 float f(float x);
14 float tolerance(float x1, float x2);
15 int secant_method (float x1, float x2, float TOL);
16
17 // main() Function
18 void main()
19 {
20     float TOL, x1, x2; //TOL = Desired Tolerance X1,X2=Initial boundary
21     printf("##### This Program is to solve a equation by Secant Method\n\n");
22     printf("Please enter Tolerance : ");
23     scanf("%f",&TOL);
24
25     START: //For Restart purpose
26
27     printf("\nEnter the lower bound of the solution :: ");
28     scanf("%f", &x1);
29     printf("\nEnter the upper bound of the solution :: ");
30     scanf("%f", &x2);
31
32     if (secant_method(x1,x2,TOL)==0) goto START;
33     else ;
34     getch();
35 }
36
37
38 // Defining function f(x)
39 float f(float x)
40 {
41     float fx = x*x*x -x*x +2; // Define f(x)
42     return fx;
43 }
44
45 //Defining Tolerance function
46 float tolerance(float x1, float x2)
47 {
48     float TOL = fabs(x2-x1);
49     return TOL;
50 }
51
52 //Defining Bisection Method
53 int secant_method (float x1, float x2, float TOL)
54 {
55     float x3;
56     int n; //n=number of iteration
57
58     if (f(x1)*f(x2)>0)
59     {

```

```

60     printf ("\nSolution doesn't exists in the domain (%f,%f)\n\n", x1,x2);
61     return (0);
62
63 }
64 else ;
65
66     for(n=1; n>0; n++)
67     {
68         x3 = x2 - (f(x2) * (x2-x1) / (f(x2)-f(x1)));
69
70         if (tolerance(x1,x2)<=TOL || fabs(f(x3))<=ERROR)
71         {
72             printf("\nSolution of the Polynomial equation is :: %f", x3);
73             printf("\nNumbner of Iteration :: %d\n\n", n);
74             break;
75         }
76         else if (f(x1)*f(x3)<0) x2=x3;
77         else if (f(x2)*f(x3)<0) x1=x3;
78         else {printf("Error!!");return 0;}
79     }
80     return 1;
81 }
82

```

Output :

This Program is to solve a equation by Secant Method

Please enter Tolerance : .001

Enter the lower bound of the solution :: -8

Enter the upper bound of the solution :: 9

Solution of the Polynomial equation is :: -0.999981

Numbner of Iteration :: 188

```

1 // Numerical Integration using Trapezoidal Rule
2 // Given Function  $f(x) = 1/(1+x^2)$ 
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <math.h>
7 #include <conio.h>
8
9 //Defining the Integral
10 float f(float x)
11 {
12     float fx= 1/(1+x*x); //  $f(x)=1/(1+x^2)$ 
13     return fx;
14 }
15
16 void trapezoidal(float xMin, float xMax, float n)
17 {
18     int i; //General purpose initializer
19     float h = (xMax-xMin)/n; // h = step size
20     float area, marginal_sections_area, middle_sections_area=0;
21
22     for(i=1; i<n-1; i++)
23     {
24         middle_sections_area = middle_sections_area + f(xMin + i*h);
25     }
26     marginal_sections_area = f(xMin) + f(xMax);
27     area = (h/2)*(marginal_sections_area + 2*middle_sections_area);
28
29     printf("Area between (%f,%f) is : %f",xMin,xMax,area);
30 }
31
32 void main()
33 {
34     printf ("## Numerical Integration using Trapezoidal Rule ##\n\n");
35
36     float xMin, xMax;
37     int n;
38
39     printf ("Please enter the lower limit of x : ");
40     scanf ("%f", &xMin);
41     printf("\nPlease enter the uppper limit of x : ");
42     scanf ("%f", &xMax);
43     printf ("\nPlease enter the total number of sections :");
44     scanf ("%d", &n);
45     printf("\n\n");
46
47     trapezoidal(xMin,xMax,n);
48
49     getch();
50 }

```


Output :

```
## Numerical Integration using Trapezoidal Rule ##
```

```
Please enter the lower limit of x : 0
```

```
Please enter the uppper limit of x : 1
```

```
Please enter the total number of sections :999
```

```
Area between (0.000000,1.000000) is : 0.784897
```

```

1 // Numerical Integration using Simpson's Rule
2 // Given Function f(x) = 1/(1+x^2)
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <math.h>
7 #include <conio.h>
8
9 //Defining the Integral
10 float f(float x)
11 {
12     float fx= 1/(1+x*x); // f(x)=1/(1+x^2)
13     return fx;
14 }
15
16 void main()
17 {
18     printf ("## Numerical Integration using Simpson's Rule ##\n\n");
19
20     float xMin, xMax, area;
21
22     printf ("Please enter the lower limit of x : ");
23     scanf ("%f", &xMin);
24     printf("\nPlease enter the uppper limit of x : ");
25     scanf ("%f", &xMax);
26
27     printf("\n\n");
28
29     // Applying Simpson's Rule
30     area = ((xMax-xMin)/6) * (f(xMin)+f(xMax)+4*f((xMin-xMax)/2));
31
32     printf("Area of the curve under (%f,%f) is :: %f",xMin,xMax,area);
33
34     getch();
35 }

```

Output :

```
## Numerical Integration using Simpson's Rule ##
```

```
Please enter the lower limit of x : 0
```

```
Please enter the uppper limit of x : 1
```

```
Area of the curve under (0.000000,1.000000) is :: 0.783333
```

```

1 // Finding the Velocity and Accleration from a given set of data of Position and Time
  by using Forward Difference Numerical Differentiation
2
3 #include<stdio.h>
4 #include<stdlib.h>
5 #include<math.h>
6 #include<conio.h>
7
8 // Function to calculate Velocity and Accleration
9 float state(int n)
10 {
11     int i; // General purpose initializer
12
13     float r[n], t[n], v[n-1], a[n-2];
14
15     printf("      Time      Position\n");
16     printf("-----      ----- \n\n");
17     for(i=0; i<=n; i++)
18     {
19         printf("      t[%d] = ",i+1);
20         scanf("%f",&t[i]);
21         printf("      r[%d] = ",i+1);
22         scanf("%f",&r[i]);
23     }
24     for(i=0; i<=n-1; i++)
25     {
26         v[i] = (r[i+1]-r[i])/(t[i+1]-t[i]);
27     }
28     for(i=0; i<=n-2; i++)
29     {
30         a[i] = (v[i+1]-v[i])/(t[i+1]-t[i]);
31     }
32
33
34     printf("      Time      Position      Velocity
35     Accleration\n");
36     printf("-----      -----      -----
37     -\n\n");
38     for(i=0; i<=n-2; i++)
39     {
40         printf("      %f      %f      %f
41         %f\n",t[i],r[i],v[i],a[i]);
42     }
43     for(i=n-2; i<=n-1; i++)
44     {
45         printf("      %f      %f      %f      \n",t[i],r[i],v[i]);
46     }
47     for(i=n-1; i<=n; i++)
48     {
49         printf("      %f      %f      \n",t[i],r[i],v[i]);
50     }
51 }
52
53 //main() Function
54 void main()
55 {
56     printf("## Finding the Velocity and Accleration from a given set of data of
57     Position and Time by using Forward Difference Numerical Differentiation ##\n\n");
58
59     int n; // n = Number of dataset

```

```
56
57 printf("Please enter the total number of datasets :");
58 scanf("%d",&n);
59 n=n-1; //Counting from 0
60
61 state(n);
62
63 printf("\n\n");
64 getch();
65 }
```

Output :

Finding the Velocity and Acceleration from a given set of data of Position and Time by using Forward Difference Numerical Differentiation

Please enter the total number of datasets :10

Time	Position		
-----	-----		
t[1] = 0			
	r[1] = 0		
t[2] = 1			
	r[2] = 1		
t[3] = 2			
	r[3] = 4		
t[4] = 3			
	r[4] = 9		
t[5] = 4			
	r[5] = 16		
t[6] = 5			
	r[6] = 25		
t[7] = 6			
	r[7] = 36		
t[8] = 7			
	r[8] = 49		
t[9] = 8			
	r[9] = 64		
t[10] = 9			
	r[10] = 81		
Time	Position	Velocity	Acceleration
-----	-----	-----	-----
0.000000	0.000000	1.000000	2.000000
1.000000	1.000000	3.000000	2.000000
2.000000	4.000000	5.000000	2.000000
3.000000	9.000000	7.000000	2.000000
4.000000	16.000000	9.000000	2.000000
5.000000	25.000000	11.000000	2.000000
6.000000	36.000000	13.000000	2.000000
7.000000	49.000000	15.000000	
8.000000	64.000000		

```

1 // Finding the Velocity and Acceleration from a given set of data of Position and Time
  by using Backward Difference Numerical Differentiation
2
3 #include<stdio.h>
4 #include<stdlib.h>
5 #include<math.h>
6 #include<conio.h>
7
8 // Function to calculate Velocity and Acceleration
9 float state(int n)
10 {
11     int i; // General purpose initializer
12
13     float r[n], t[n], v[n], a[n];
14
15     n=n-1; //Counting from 0
16
17     printf("      Time      Position\n");
18     printf("-----\n\n");
19     for(i=0; i<=n; i++)
20     {
21         printf("      t[%d] = ",i+1);
22         scanf("%f",&t[i]);
23         printf("      r[%d] = ",i+1);
24         scanf("%f",&r[i]);
25     }
26     //Calculating v[n]
27     for(i=n; i>=1; i--)
28     {
29         v[i] = (r[i-1]-r[i])/(t[i-1]-t[i]);
30     }
31     //Calculating a[n]
32     for(i=n; i>=2; i--)
33     {
34         a[i] = (v[i-1]-v[i])/(t[i-1]-t[i]);
35     }
36
37
38     printf("      Time      Position      Velocity
  Acceleration\n");
39     printf("-----\n\n");
40     for(i=0; i<1; i++)
41     {
42         printf("      %f      %f      \n",t[i],r[i],v[i]);
43     }
44     for(i=1; i<2; i++)
45     {
46         printf("      %f      %f      %f      \n",t[i],r[i],v[i]);
47     }
48     for(i=2; i<n; i++)
49     {
50         printf("      %f      %f      %f
  %f\n",t[i],r[i],v[i],a[i]);
51     }
52 }
53
54 //main() Function
55 void main()
56 {

```

```
57     printf("## Finding the Velocity and Accleration from a given set of data of  
Position and Time by using Backward Difference Numerical Differentiation ##\n\n");  
58  
59     int n; // n = Number of dataset  
60  
61     printf("Please enter the total number of datasets :");  
62     scanf("%d",&n);  
63  
64     state(n);  
65  
66     printf("\n\n");  
67     getch();  
68 }
```


Output :

Finding the Velocity and Acceleration from a given set of data of Position and Time by using Backward Difference Numerical Differentiation

Please enter the total number of datasets :10

Time	Position		
-----	-----		
t[1] = 0	r[1] = 0		
t[2] = 1	r[2] = 2		
t[3] = 2	r[3] = 4		
t[4] = 3	r[4] = 9		
t[5] = 4	r[5] = 16		
t[6] = 5	r[6] = 25		
t[7] = 6	r[7] = 36		
t[8] = 7	r[8] = 37		
t[9] = 8	r[9] = 64		
t[10] = 9	r[10] = 81		
Time	Position	Velocity	Acceleration
-----	-----	-----	-----
0.000000	0.000000		
1.000000	2.000000	2.000000	
2.000000	4.000000	2.000000	-0.000000
3.000000	9.000000	5.000000	3.000000
4.000000	16.000000	7.000000	2.000000
5.000000	25.000000	9.000000	2.000000
6.000000	36.000000	11.000000	2.000000
7.000000	37.000000	1.000000	-10.000000
8.000000	64.000000	27.000000	26.000000

```

1 // Solving a ODE by using Euler's Method
2 // Given function f'(x) = 3x^2 +1
3
4 #include<stdio.h>
5 #include<conio.h>
6
7 float euler_int_soln(float xi, float yi, float xf, float h);
8
9 float dYdX (float x)
10 {
11     float dYdX = 3*x*x +1;
12     return dYdX;
13 }
14
15 void main()
16 {
17     printf("## Solving first ordinary differential equation by using Euler's Method\n\n");
18
19     float xi, yi, xf, h;
20
21     printf("Please Enter the ititial value conditions\n");
22     printf("Initial valu of x >>");
23     scanf("%f",&xi);
24     printf("Initial value of y at x=%f >>",xi);
25     scanf("%f",&yi);
26     printf("\nPlease Enter the Final value of x\n>>");
27     scanf("%f",&xf);
28     printf("Please Enter the step size >>");
29     scanf("%f",&h);
30
31     euler_int_soln(xi,yi,xf,h);
32
33     getch();
34 }
35
36 //Defining Euler's Method of solution
37 float euler_int_soln(float xi, float yi, float xf, float h)
38 {
39     float x, y;
40
41     x=xi;
42     y=yi;
43
44     for(x; x<=xf; x=x+h)
45     {
46         y = y + h*dYdX(x);
47     }
48
49     printf("\nSolution for y at x=%f is :: %f",x,y);
50
51     return 0;
52 }

```

Output :

```
## Solving first ordinary differential equation by using Euler's Method ##
```

```
Please Enter the ititial value conditions
```

```
Initial valu of x >>0
```

```
Initial value of y at x=0.000000 >>3
```

```
Please Enter the Final value of x
```

```
>>9
```

```
Please Enter the step size >>.0001
```

```
Solution for y at x=9.000046 is :: 740.148132
```