CHAPTER IV

C H A P T E R - IV

COMPUTER IMPLEMENTATION OF SOME METHODS

4.1 INTRODUCTION

In this chapter, computer programs of the following integration methods are given in C language.

(a) Cubic spline integration method.

(b) Integration using trapezoidal rule.

(c) Integration using Simpson's first rule.

(d) Integration using Gauss-Legendre method.

The listing of these programs are given in the next section. Numerical integration of four functions are evaluated using these programs. The outputs of these programs are given and compared with their true answer in the last section of this chapter.

Here, we shall consider the integration of $x^{1/2}$, x^2, $1/(1+x)$, $1/(1+x^2)$. For the cubic spline integration method, we consider data at 10 non-equally spaced base-points in the interval 0 to 1 as given in the following table.

Base points	function values			
	$x^{1/2}$	x^2	$1/(1+x)$	$1/(1+x^2)$
0.00	0.0000000	0.0000	1.0000000	1.0000000
0.15	0.3872983	0.0225	0.8695652	0.9779951
0.25	0.5000000	0.0625	0.8000000	0.9411764
0.30	0.5477225	0.0900	0.7692307	0.9174311
0.45	0.6708203	0.2025	0.6896551	0.8316008
0.62	0.7874007	0.3844	0.6172839	0.7223345
0.76	0.8717797	0.5776	0.5681818	0.6338742
0.82	0.9055385	0.6724	0.5494505	0.5979430
0.92	0.9591663	0.8464	0.5208333	0.5415944
1.00	1.0000000	1.0000	0.5000000	0.5000000

The computer program for cubic spline integration method is based on natural spline condition, and it uses the equation (2.33A). Also, this program uses pointers in C Language.

For the remaining three computer programs, the lower and upper limits of integration are the input values. As an example, the function $x^{1/2}$ is considered in the listing of all programs. For the testing of convergence the constant epsilon (EPS) is defined in the program for trapezoidal as well as composite Simpson's first rule. Moreover, the program for composite Simpson's first rule takes the value for maximum number of iterations to exit from the loop in finite time, while in the program for trapezoidal rule, we have defined this maximum number by

constant, JMAX. Here the subroutine 'trapzd' is used in the subroutine 'strap' which is used further in the main program.

The fourth computer program is prepared for the ten point Gauss-Legendre formula. In this program, two arrays x[] and w[] are used. These arrays store the values of abscissas and weights respectively corresponding to the ten point Gauss-Legendre formula. This program is based on the equation (3.31). Therefore, the standard interval [-1,1] is automatically transformed to the desired interval.

4.2 PROGRAM LISTING

Cubic spline integration rule:

```
/* This is the main program. In this program we get
integration of the      function given in the  tabulated  form
using spline function.    */
#include <stdio.h>
#include <math.h>
#include "nrutil.h"
#include "nrutil.c"
main()
{
 int n,i;
 float *a,*b,*c,*d,*x,*y,*m,*h,*u,sum1,sum2,sum,term1,term2;
 void tridag(float *a, float *b, float *c, float *d,
                  float *m,int n);
     /* Read in a value for n */
 printf("\n How many tabulated values are there ? ");
 scanf("%d", &n);
```

```
n -= 1;
  /*   Allocate memory  */
a=vector(Ø,n);              /*(int *)malloc(n*sizeof(float));*/
b=vector(Ø,n);
c=vector(Ø,n);
d=vector(Ø,n);
x=vector(Ø,n);
y=vector(Ø,n);
m=vector(Ø,n);
h=vector(Ø,n);
u=vector(Ø,n);
    /*  Read in the values from tables   */
printf("\n Enter the values of X array one by one.");
for( i = Ø; i <= n ; i++ )
     scanf("%f", x+i );
printf("\n Enter the values of Y array one by one.");
for( i = Ø; i <= n ; i++ )
      scanf("%f", y+i );
for( i=1; i <= n; i++ )
      *(h+i)= *(x+i)-(*(x+i-1));
*a=Ø;
for( i=1; i < n; i++ )
      *(a+i) = *(h+i)/(*(h+i)+(*(h+i+1)));
for( i=Ø; i <= n; i++ )
      *(u+i) = 2.Ø;
for( i=1; i < n; i++ )
      *(b+i) = 1.Ø - *(a+i);
*(b+n)=Ø;
for( i=1; i <= n; i++ )
      *(c+i) = (*(y+i)/(*(h+i)) - (*(y+i-1))/(*(h+i)));
```

```
for( i=1; i < n; i++ )
      *(d+i) =((*(c+i+1)-(*(c+i)))/(*(h+i)+(*(h+i+1))))*6.Ø;
free_vector(c,1,n);
        /* Assuming the spline as natural cubic spline  we
assign thevalue zero  for  the remaining elements of array
a[],b[],d[].*/
*(a+n) = Ø.Ø;
*b=Ø.Ø;
*d=Ø.Ø;
*(d+n)=Ø.Ø;
  /*  Solve the tridiagonal mmatrix for  m[Ø..n]   */
tridag(a,u,b,d,m,n);
for( i=Ø; i <= n; i++ )
       printf("\n m[%d] =  %f ", i, *(m+i) );
sum1=sum2=Ø.Ø;
term1=term2=Ø.Ø;
for( i=1; i <= n; i++ )    {
     term1 =((*(y+i-1)/2.Ø)+(*(y+i)/2.Ø))*(*(h+i));
     term2 =((*(m+i-1)/24.Ø)+(*(m+i)/24.Ø))*pow(*(h+i),3);
     sum1 += term1;
     sum2 += term2;
}
sum = sum1 - sum2;
printf("\n  The value of the integration is : %f \n" , sum);
}

void tridag(float *a, float *b, float *c, float *r,
                                    float *u,int n)
   /* This function solves the tridigonal linear set of
```

```
equations. Vectors a[Ø..n], b[Ø..n], c[Ø..n] and r[Ø..n]
are inputs for this function .  */
 {
    int j;
    float bet,*temp;
    temp=vector(Ø,n);
        /* One vector of workspace, temp is needed.*/
    if  (b[Ø] == Ø.Ø) nrerror("Error 1 in tridag");
        /* If this happens then rewrite set of equations
           as a set of order N-1.*/
    u[Ø] = r[Ø]/(bet=b[Ø]);
    for ( j=1; j<=n; j++) {   /*  Decomposition and forward
                                          substitution. */
         *(temp+j) = *(c+j-1)/bet;
         bet=*(b+j)-(*(a+j))*(*(temp+j));
         if (bet == Ø.Ø)   nrerror("Error 2 in tridag");
                  /*Algorithm fails.*/
         *(u+j)=(*(r+j)-(*(a+j))*(*(u+j-1)))/bet;
    }
    for (j = (n-1); j >= Ø; j--)
        *(u+j) -= *(temp+j+1)*(*(u+j+1));
                                /* Backsubstitution  */
    free_vector(temp,Ø,n);
    return;
 }
```

Trapezoidal Rule Program :

```
#include <stdio.h>
#include <math.h>
#define  EPS  1.0e-5
#define  JMAX  40
main()
{
  int n,i;
  float a, b, olds, s;
  float func(float x);
  float strap(float (func)(float), float a, float b);
  printf("\n Give the initial and final values of the
              interval of integration.");
  printf("\n a= ");
  scanf("%f", &a);
  printf("\n b= ");
  scanf("%f", &b);
  s = strap( func, a, b );
  printf("\n The value of the integration is = %f\n", s);
}
  float strap(float (*func)(float), float a, float b)
  {
    float trapzd(float (* func)(float), float a, float b,
                                        int n);
    int i;
    float sumint, olds;
    olds = -1.0e30;
    for( i=1; i<=JMAX; i++)    {
      sumint = trapzd( func, a, b, i);
```

```
    if(fabs(sumint-olds) < EPS*fabs(olds))  return sumint;
    olds = sumint;
   }
   printf("\nToo many steps in routine strap .");
   return 0.0;          /* never get here. */
 }

float trapzd(float (*func)(float), float a, float b, int n)
 {
   float x, tnm, sum, del;
   static float s;
   int it , j;
   if (n==1) {
       return(s=0.5*(b-a)*(func(a) + func(b)));

   } else {
        for (it=1,j=1; j<n-1; j++) it <<= 1;
        tnm=it;
        del=(b-a)/tnm; /* This is the spacing to be added. */
        x=a+0.5*del;
        for ( sum=0.0,j=1; j<=it; j++,x+=del) sum += func(x);
        s=0.5*(s+(b-a)*sum/tnm);
        return(s);
    }
 }
  float  func(float x)
  {
         float y;
         y = pow(x,0.5);
         return(y);
  }
```

Simpson's 1/3rd Rule

```
#include <stdio.h>
#include <math.h>
#define  EPS  1.Øe-5
main()
{
   int  n,i,it,maxit;
   double x,curs,olds,inits,evens,odds,h;
   float a,b;
   double func( float u);
   printf(" \n GIVE THE LOWER & UPPER LIMITS OF INTEGRATION.---");
   printf("\n a= ");
   scanf("%f",&a);
   printf("\n b= ");
   scanf("%f",&b);
   printf("\n GIVE MAXIMUM NUMBER OF ITERATIONS.----");
   scanf("%d",&maxit);
   n=1;
   it=Ø;
   curs=Ø.Ø;
   olds=Ø.Ø;
   evens=Ø.Ø;
   odds=Ø.Ø;
   inits=func(a)+func(b);
   printf("\n   ITERATION NO      INTEGRAL  VALUE      ");
   printf("\n   ----------------------------------------");
   for( it=1; it<maxit; it++) {
        h=(b-a)*Ø.5/n;
        odds=Ø.Ø;      /* Calculation of odd sums     */
```

```
        x=a+h;
        for( i=1; i<=n; i++ ) {
                odds=odds+func(x);
                x=x+2*h;
        }
        curs=(inits+4*odds+2*evens)*h/3;
        if(fabs(curs-olds) < EPS )goto end;
        printf("\n %d  %lf", it,curs);
        olds=curs;
        evens += odds;
        n=2*n;
        it += 1;
    }
    end:
    ;
}
  double func(float u)
  {
        float y;
        y=pow(u,Ø.5);
        return(y);
  }
```

Gauss-Legendre Integration

```
#include <stdio.h>
#include <math.h>
main()
{
  int i;
  float a,b;
  static double sum,sumint,ts,td,tr;
  static float x[]={0.1488743389, 0.4333953941, 0.6794095682,
            0.8650633666,0.9739065285, -0.1488743389,
            -0.4333953941, -0.6794095682,
             -0.8650633666, -0.9739065285};
  static float w[]={0.2955242247,0.2692667193,0.2190863625,
            0.1494513491,0.0666713443,0.2955242247,
            0.2692667193, 0.2190863625,
            0.1494513491,0.0666713443};
  double func( float x);
  printf("Give the lower & upper limit of the integration.\n");

  scanf("%f %f",&a,&b);
  sum=0;
  ts=(b+a)*0.5;
  td=(b-a)*0.5;
  for(i=0; i<10; i++){
        tr=x[i]*td+ts;
        sum += w[i]*func(tr);
  }
  sumint=td*sum;
  printf(" The value of the integration is: %lf", sumint );
}
```

```
double func( float x)
{
  float y;
    y=1/x;
    return(y);
}
```

4.3 OUTPUT OF PROGRAMS

The output of the spline integration program is as follows:

Function	Computed value	Exact value
$X^{1/2}$	Ø.659945	Ø.6666667
X^{2}	Ø.333523	Ø.3333333
1/(1+x)	Ø.693314	Ø.69314718
$1/(1+x^2)$	Ø.785228	Ø.7857142

The output for the trapezoidal rule is taken for two different epsilon values viz 1.Øe-3 and 1.Øe-5. The results are given below.

Function	Eps = 1.Øe-3	Eps = 1.Øe-5	Exact Value
$x^{1/2}$	Ø.666526	Ø.666664	Ø.6666667
x^{2}	Ø.333374	Ø.333334	Ø.3333333
1/(1+x)	Ø.693208	Ø.693148	Ø.69314718
$1/(1+x^2)$	Ø.785235	Ø.785396	Ø.7857142

The outputs for the Simpson's rule program are taken for 20 iterations. The results are given below.

Function	Iteration Number	Integral Value
$X^{1/2}$	1	0.638071
	3	0.656526
	5	0.663079
	7	0.665398
	9	0.666218
	11	0.666508
	13	0.666619
	15	0.666647
	17	0.666660
X^{2}	1	0.333333
$1/(1+x)$	1	0.694444
	3	0.693254
	5	0.693155
$1/(1+x^{2})$	1	0.783333
	3	0.785392

The outputs for Gauss-Legendre formula is as follows:

Functions	$X^{1/2}$	X^2	$1/(1+x)$	$1/(1+x^8)$
Computed Value :	Ø.666756	Ø.333333	Ø.693147	Ø.785398
True Value:	Ø.6666667	Ø.333333	Ø.69314718	Ø.7857142

4.4 CONCLUSION

Here we discuss some general behaviour of the computer programs. The programs developed here are applicable to a class of continuous functions. It can be observed from the outputs given above that the accuracy of the calculated results can be improved by chossing epsilon as small as required. Program for Simpson's 1/3 rule can give more accuracy if we increase the number of iterations. Also, if we increase the number of base points in the cubic spline integration program then it will give more accurate results. Here we observed that program for 1Ø- point Gauss-Legendre formula also gives more accuracy for some functions.