ICT Tool: - 'C' Language Program for Shooting Method

P R Kolhe, M H Tharkar, P P Kolhe N Mirajkar
Associate Professor (CAS). CAET Dapoli, India.

Abstract – In numerical analysis, the shooting method is a method for solving a boundary value problem by reducing it to the solution of an initial value problem. Roughly speaking, we 'shoot' out trajectories in different directions until we find a trajectory that has the desired boundary value. In the era of Information Communication Technology (ICT) .The ICT programming technique, it is easier task. One of the very popular programs in C programming is Shooting Method. This paper discuss Shooting Method in C language, source code and methods with outputs. The source codes of program for Shooting Method in C programming are to be compiled. Running them on Turbo C or available version and other platforms might require a few modifications to the code.

Index Terms – Shooting Method, ICT,C Lang., Turbo C.

Introduction to Shooting Method

Shooting method is a famous method for numerical solution of second order differential equation when boundary condition is known. In this tutorial, we're going to write a program for **Shooting method in C** with sample output and working procedure of the method.

Shooting Method is stated as

For a boundary value problem of a second-order ordinary differential equation, the method is stated as follows.

$$y''(t) = f(t, y(t), y'(t)), \quad y(t_0) = y_0, \quad y(t_1) = y_1$$

Let be the boundary value problem. Let $y(t; a)$ denote the solution of the initial value problem

$$y''(t) = f(t, y(t), y'(t)), \quad y(t_0) = y_0, \quad y'(t_0) = a$$

Define the function $F(a)$ as the difference between $y(t_1; a)$ and the specified boundary value y_1.

$$F(a) = y(t_1; a) - y_1$$

If F has a root a then the solution $y(t; a)$ of the corresponding initial value problem is also a solution of the boundary value problem. Conversely, if the boundary value problem has a solution $y(t)$, then $y(t)$ is also the unique solution $y(t; a)$ of the initial value problem where $a = y'(t_0)$, thus a is a root of F.

Procedure of C program for shooting method is given below:

1. As the user executes the program, it asks for boundary values i.e. initial value of x (x_0), initial value of y (y_0), final value of x (x_n), final value of y (y_n) and the value of increment (h).
2. The second step of calculation is to convert this boundary value problem into initial value problem.
3. After the conversion into initial value problem, the user has to input the initial guess value of z (M1) which is known as shooting.
4. Using this guess value of z, the program calculates intermediate values of z & y are calculated. The final value of y obtained is assigned as B1 in the source code.
5. Again, the user has to shoot i.e. the shooting method program asks second initial guess value of z (M2).
6. Using M2, new values of y and z are calculated. The final value of y obtained in second guess is assigned as B2 in the program.
7. (M1, B1), (M2, B2), and (y0, z0) are assumed to be collinear in this C program and value of z0 determined using following equation, (B2-B1)/(M2-M1) = (z0-B2)/(y0-M2)
8. Finally, the program prints the result upto 6 decimal places.

C Program for Shooting Method

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
float f1(float x,float y,float z)
{
  return(z);
}
float f2(float x,float y,float z)
{
```

```
    return(x+y);
}
float shoot(float x0,float y0,float z0,float xn,float h,int p)
{
    float x,y,z,k1,k2,k3,k4,l1,l2,l3,l4,k,l,x1,y1,z1;
    x=x0;
    y=y0;
    z=z0;
    do
    {
        k1=h*f1(x,y,z);
        l1=h*f2(x,y,z);
        k2=h*f1(x+h/2.0,y+k1/2.0,z+l1/2.0);
        l2=h*f2(x+h/2.0,y+k1/2.0,z+l1/2.0);
        k3=h*f1(x+h/2.0,y+k2/2.0,z+l2/2.0);
        l3=h*f2(x+h/2.0,y+k2/2.0,z+l2/2.0);
        k4=h*f1(x+h,y+k3,z+l3);
        l4=h*f2(x+h,y+k3,z+l3);
        l=1/6.0*(l1+2*l2+2*l3+l4);
        k=1/6.0*(k1+2*k2+2*k3+k4);
        y1=y+k;
        x1=x+h;
        z1=z+l;
        x=x1;
        y=y1;
        z=z1;
        if(p==1)
        {
            printf("\n%f\t%f",x,y);
        }
    }while(x<xn);
    return(y);
}
 main()
{
    float x0,y0,h,xn,yn,z0,m1,m2,m3,b,b1,b2,b3,e;
    int p=0;
    printf("\n  Enter x0,y0,xn,yn,h:");
    scanf("%f%f%f%f%f",&x0,&y0,&xn,&yn,&h);
    printf("\n  Enter the trial M1:");
    scanf("%f",&m1);
    b=yn;
    z0=m1;
    b1=shoot(x0,y0,z0,xn,h,p=1);
    printf("\nB1 is %f",b1);
    if(fabs(b1-b)<0.00005)
    {
        printf("\n  The value of x and respective z are:\n");
        e=shoot(x0,y0,z0,xn,h,p=1);
        return(0);
    }
    else
    {
    printf("\nEnter the value of M2:");
    scanf("%f",&m2);
    z0=m2;
    b2=shoot(x0,y0,z0,xn,h,p=1);
    printf("\nB2 is %f",b2);
    }
    if(fabs(b2-b)<0.00005)
    {
         printf("\n  The value of x and respective z are\n");
         e= shoot(x0,y0,z0,xn,h,p=1);
         return(0);
    }
    else
    {
        printf("\nM2=%f\tM1=%f",m2,m1);
        m3=m2+(((m2-m1)*(b-b2))/(1.0*(b2-b1)));
        if(b1-b2==0)
        exit(0);

        printf("\nExact value of M =%f",m3);
        z0=m3;
        b3=shoot(x0,y0,z0,xn,h,p=0);
    }
```

```
if(fabs(b3-b)<0.000005)
{
   printf("\nThere is solution :\n");
   e=shoot(x0,y0,z0,xn,h,p=1);
   exit(0);
}
  do
  {
     m1=m2;
     m2=m3;
     b1=b2;
     b2=b3;
     m3=m2+(((m2-m1)*(b-b2))/(1.0*(b2-b1)));
     z0=m3;
     b3=shoot(x0,y0,z0,xn,h,p=0);

  }while(fabs(b3-b)<0.0005);
  z0=m3;
  e=shoot(x0,y0,z0,xn,h,p=1);
 }
```

Output of Shooting Method

```
Enter x0,y0,xn,yn,h: 0 1 3 4 0.5
Enter the trial M1:0

0.500000        1.148438
1.000000        1.717346
1.500000        2.979375
2.000000        5.383970
2.500000        9.672014
3.000000        17.064804
B1 is 17.064804
Enter the value of M2:1

0.500000        1.669271
1.000000        2.891934
1.500000        5.107366
2.000000        9.008181
2.500000        15.716896
3.000000        27.072252
B2 is 27.072252
M2=1.000000     M1=0.000000
Exact value of M =-1.305508
There is solution :

0.500000        0.468485
1.000000        0.183913
1.500000        0.201267
2.000000        0.652535
2.500000        1.780370
3.000000        3.999998
Process returned 0 (0x0)   execution time : 26.508 s
Press any key to continue.
```

REFERENCES

[1] Josef Stoer and Roland Bulirsch. Introduction to Numerical Analysis. New York: Springer-Verlag, 1980. (See Section 7.3.) Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007).
[2] *Computer Graphics by Chennakesava Alavala*
[3] Computer Graphics by Steven Harrington
[4] Soil and Water Conservation Engineering by Dr. R. Suresh
[5] E Balagurusamy, 1989 Programming in ANSI C (4E) Tata Mc Graw-Hill Publisher, New Delhi
[6] *Y. Kanetkar Lets C*