A Review of Numerical Techniques of Solving Ordanary Differential Equations using C and C++ Languages.

Rahul Solanki[1]
Department Of Mathematics, Jodhpur Institute Of Enginering And Technology,
Jodhpur, Rajasthan.

Rakshak Solanki[2]
Department Of Applied Sciences, I-Year, Jodhpur Institute Of Enginering And Technology,
Jodhpur, Rajasthan.

Abstract: **Ordinary differential equations (ODEs) are basically involved in many engineering problems. Many applications of differential equation are useful to solve problems like heat-time relation, velocity-time relation, traffic transportation problems etc. They arise in models throughout mathematics, science, and engineering. By itself, a system of ODEs has many solutions. Some systems work with initial values called Initial Value Problems. However, in many applications a solution is determined in a more complicated way. A boundary value problem (BVP) specifies values or equations for solution components at more than one variable. Unlike IVPs, a boundary value problem may not have a solution, or may have a finite number of solution or infinitely many solutions. Some problems are so complex that we need computer programme to solve them so programs are developed for solving BVPs require users to provide a guess for the solution desired. Often there are parameters that have to be determined so that the BVP has a solution. Again there might be more than one possibility, so programs require a guess for the parameters desired. Many problems, arising in a wide variety of application areas, give rise to mathematical models which form boundary value problems for ordinary differential equations. These problems rarely have a closed form solution, and computer simulation is typically used to obtain their approximate solution.**

The study being carried out is based on the investigation of the numerical methods for solving first-order ordinary differential equations. The aim of the study is to give a comparative analysis of the numerical methods for solving first-order differential equations by use of an alternative approach which is the use of numerical approximation methods to find an accurate approximation to the desired solution of an initial value problem. The methods are presented in the simplest context possible which is a single scalar first order differential equation.

AMS 2010 Subject Classification: **65D, 65F.**

Key Words: ODE, LDE, R-K 4th order method, Milne's P-C method, Euler's method.

1. INTRODUCTION

Numerical analysis is the study of algorithms that use numerical approximation for the problems of mathematical analysis. Numerical analysis naturally finds applications in all fields of engineering and the physical sciences, but in the 21st century also the life sciences and even the arts have adopted elements of scientific computations. Ordinary differential equations appear in celestial mechanics (planets, stars and galaxies); numerical linear algebra is important for data analysis; stochastic differential equations and Markov chains are essential in simulating living cells for medicine and biology. When using numerical methods or algorithms and computing with finite precision, errors of approximation or rounding and truncation are introduced. In numerical analysis, Numerical integration constitutes a broad family of algorithms for calculating the numerical value of a definite integral, and by extension, the term is also sometimes used to describe the numerical solution of differential equations. This article focuses on calculation of definite integrals. The term numerical quadrature (often abbreviated to *quadrature*) is more or less a synonym for *numerical integration*, especially as applied to one-dimensional integrals. Numerical integration over more than one dimension is sometimes incorrectly described as cubature,[1] since the meaning of *quadrature* is understood for higher-dimensional integration as well. Many differential equations cannot be solved using symbolic computation ("analysis"). For practical purposes, however – such as in engineering – a numeric approximation to the solution is often sufficient. The algorithms studied here can be used to compute such an approximation. An alternative method is to use techniques from calculus to obtain a series expansion of the solution.

Ordinary differential equations occur in many scientific disciplines, for instance in physics, chemistry, biology, and economics. In addition, some methods in numerical partial differential equations convert the partial differential equation into an ordinary differential equation, which must then be solved.

2. IMPLEMENTATION OF C++ PROGRAM IN SOLVING ORDINARY LINEAR DIFFERENTIAL EQUATION.

Numerical methods help us to find a better solution of any problems. In engineering, we face practical problems and the solution of those problems can only be obtain using numerical methods. In advanced technology, we use computer programming to solve ODE. Here we are going to use C++ programming to solve a LDE using Runga-Kutta 4th order method, Milne's Predictor & Corrector method and Euler's method.

2.1. Algorithm 1

Euler's Method

```
#include<stdio.h>
#include<conio.h>
int fn;
float a,b,x,y,h,t,k,q;
float fun(float x,float y)
{
  switch(fn)
        {
              case 1 : return ((3*x)+(y/2));
              case 2 : return ((y-x)/(y+x));
              case 3 : return (y-2*x/y);
              case 4 : return ((x*x+1)*(y*y)/2);
              case 5 : return (3*exp(x)+2*y);
          // default: cout<<"Enter a Valid Choice: ";
        }
}
void main()
{
  clrscr();
  printf("Select a function (1-5) from following:\n1. dy/dx=3x+y/2");
  printf("\n2. dy/dx=(y-x)/(y+x)");
  printf("\n3. dy/dx=y-2x/y");
  printf("\n4. dy/dx=(1/2)(x^2+1)(y^2)") ;
  printf("\n5. dy/dx=3e^x+2y\n");
  scanf("%d",&fn);
  printf("\nEnter x(0) : ");
  scanf("%f",&a);
  printf("\nEnter y(0) : ");
  scanf("%f",&b);
  printf("\nEnter step size (h) : ");
  scanf("%f",&h);
  printf("\nEnter end point (xn) : ");
  scanf("%f",&t);
  x=a;
  y=b;
  clrscr();
  printf("\n\t X\t Y\n");
  while(x<t)
  {
    k=h*fun(x,y);
    y=y+k;
    x=x+h;
    printf("\t%0.3f\t%0.3f\n",x,y);
  }
  scanf("%f",&q);
}
```

2.2 Algorithm 2

Milne's Predictor & Corrector Method.

```
#include<iostream.h>
#include<conio.h>
#include<math.h>

float n,xf,h,x[150],y[150],f[150];
int fn;
void input();
void milne();
void output();
float function(float, float);
void select(void);

void main()
{
 clrscr();
 select();
 input();
 milne();
 clrscr();
 output();
 getch();
}

void input()
{    cout<<"\nEnter value of x(0): " ;
cin>>x[0];
cout<<"\nEnter value of y(0): ";
```

```
    cin>>y[0];
    cout<<"\nEnter value of xf: ";
    cin>>xf;
    cout<<"\nEnter value of number of iteration (n): ";
    cin>>n;

}
```

2.3 Algorithm 3

Runga-Kutta 4th order Method.

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
int fn;
float f(float x,float y);

void main()
 {
   clrscr();
   float x0,y0,k1,k2,k3,k4,k,y,x,h,xn,u,n;
   cout<<"Select a function (1-5) from following:\n1.
dy/dx=3x+y/2";
   cout<<"\n2. dy/dx=(y-x)/(y+x)";
   cout<<"\n3. dy/dx=y-2x/y";
   cout<<"\n4. dy/dx=(1/2)(x^2+1)(y^2)" ;
   cout<<"\n5. dy/dx=3e^x+2y\n";
   cin>>fn;
   cout<<"\n\nEnter x0 : ";
   cin>>x0;
   cout<<"\n\nEnter y0 : ";
   cin>>y0;
   cout<<"\n\nEnter end point (xn) : ";
   cin>>xn;
   cout<<"\n\nEnter step size (h) : ";
   cin>>h;
   x=x0;
   y=y0;
   clrscr();
   cout<<"\n\nX\t\tY\n";
   n=(xn-x0)/h;
    for(int i=0; i<=n; i++)
    {
  k1=h*f(x,y);
  k2=h*f((x+h/2),(y+k1/2));
  k3=h*f((x+h/2),(y+k2/2));
  k4=h*f((x+h),(y+k3));
  k=((k1+k2+k2+k3+k3+k4)/6);
  cout<<"\n"<<x<<"\t\t"<<y;
  y=y+k;
  x=x+h;

    }
    getch();

}
float f(float x,float y)
{
     switch(fn)
          {
               case 1 : return ((3*x)+(y/2));
               case 2 : return ((y-x)/(y+x));
               case 3 : return (y-2*x/y);
               case 4 : return ((x*x+1)*(y*y)/2);
               case 5 : return (3*exp(x)+2*y);
          // default: cout<<"Enter a Valid
Choice: ";
          }
}
```

3. WHY NUMERICAL METHODS?

Basically Numerical method are key of every practical problem. We can reach to a particular solution of every practical problem using numerical methods. Generally every wave equation, heat equation, Laplace equation, equation of motions deal with partial differential equation and using some parameters we can convert them into ordinary differential equations. After converting them into ordinary differential equation, numerical method is the best tool to solve them, to compare those equations with some other previous results, make graphs to compare them.
Hence to obtain solution of any differential equation for given boundary conditions, we must use numerical methods.

4. CONCLUSION

1. Generally many engineering problems associated with ordinary differential equations or partial differential equations.

2. Solutions of these differential equations can be obtained using analytical methods.

3. Where analytical methods failed, we use numerical techniques to solve them.

4. As there are so many numerical techniques available, we can use any of one and can obtain solution of differential equation using given boundary conditions.

5. C^{++} is an important technique to solve mathematical problems, so we use it to solve ordinary differential by Euler's method, Milne's P-C method and R-K 4^{th} order method.

5. REFERENCES

[1] Corbit, D. "Numerical Integration: From Trapezoids to RMS: Object-Oriented Numerical Integration." *Dr. Dobb's J.*, No. 252, 117-120, Oct. 1996.
[2] Davis, P. J. and Rabinowitz, P. *Methods of Numerical Integration, 2nd ed.* New York: Academic Press, 1984.
[3] Hildebrand, F. B. *Introduction to Numerical Analysis.* New York: McGraw-Hill, pp. 319-323, 1956.
[4] Krommer, A. R. and Ueberhuber, C. W. *Numerical Integration on Advanced Computer Systems.* Berlin: Springer-Verlag, 1994.
[5] Milne, W. E. *Numerical Calculus: Approximations, Interpolation, Finite Differences, Numerical Integration and Curve Fitting.* Princeton, NJ: Princeton University Press, 1949.
[6] Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; and Vetterling, W. T. *Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed.* Cambridge, England: Cambridge University Press, 1992.
[7] Smith, J. M. "Recent Developments in Numerical Integration." *J. Dynam. Sys., Measurement and Control* **96**, 61-70, Mar. 1974.
[8] Ueberhuber, C. W. "Numerical Integration." Ch. 12 in *Numerical Computation 2: Methods, Software, and Analysis.* Berlin: Springer-Verlag, pp. 65-169, 1997.
[9] Weisstein, E. W. "Books about Numerical Methods." http://www.ericweisstein.com/encyclopedias/books/NumericalMethods.html.
[10] Whittaker, E. T. and Robinson, G. "Numerical Integration and Summation." Ch. 7 in *The Calculus of Observations: A Treatise on Numerical Mathematics, 4th ed.* New York: Dover, pp. 132-163, 1967.
[11] Wikkipedia.