RESEARCH ARTICLE OPEN ACCESS

Sage Package for Symbolic Computation of Series Solution of Ordinary Differential Equations

M.Kaliyappan*, A.Vanav Kumar**, S.Hariharan***
*(School of Advanced Science, VIT University, Chennai)
** (School of Advanced Science, VIT University, Chennai)
*** (School of Advanced Science, VIT University, Chennai)

ABSTRACT
The ordinary linear differential equations with constant coefficients can be solved by the algebraic methods and the solutions are obtained by elementary functions called the fundamental solutions. Most of the differential equations met in mathematics, physics and engineering sciences remain out of this class. In such cases, it is natural to search the solutions in the form of infinite series. The power series method is a well known procedure to solve ODEs with variable coefficients. The resulting series can be used to analyze and visualize the nature of the solutions of ODE's for which direct calculation is difficult. Manually solving a series solution using power series method is tedious and time consuming. In this paper a special package using SAGE (an open source software) have been developed, which can calculate any number of coefficients with no restriction to obtain the series solutions of up to third order ODE at an ordinary point. Also the well known Legendre and Hermite polynomials of any order can be generated and visualized by the way getting series solutions, In addition, Frobenius method for finding series solution around regular singular point cases are discussed.
Keywords **-** Ordinary Differential Equations, Power Series Solutions, SAGE for Differential Equations

I. INTRODUCTION

Many differential equations arising in practical applications are linear with variable coefficients which are not reduced by a differential equation with constant coefficients, needs a method to obtain classical solutions. Power series method is a standard strategy to solve linear differential equations with variable coefficients. The procedure to obtain power series solutions are well known and it is discussed in many differential equations book and some of the advanced engineering Mathematics books[1,2,4,5]. Manually solving these kind of equations can be tedious and in order to get more coefficients of the series it takes lot more time and mainly it is not possible to visualize the solutions without the existing mathematical packages.

In this paper, a package through SAGE, to obtain the series solutions of second order ODE with ordinary point and Frobenius method for the case of solutions of the indicial equations not differ by integers through symbolic computation have been developed. The Legendre and Hermite differential equations have been solved and obtained their polynomials from their series solutions. Section II describes theoretical structure of solving second order ODE through power series method. A package through SAGE using symbolic computation have been given in Section III in order to solve third order ordinary differential equations using series solution. Section IV & V describes the Legendre and Hermite differential equations and visualizing their polynomials respectively. The method of Frobenius for regular singular points are discussed in Section VI. Section VII describes the computation time comparison with MATLAB.

II. SERIES SOLUTION FOR SECOND ORDER ODE HAVING VARIABLE COEFFICIENTS

Consider the linear second-order differential equation

$$a_2(x)y'' + a_1(x)y' + a_0(x)y = 0 \ldots \quad (1)$$

which can be converted to the following standard form

$$y'' + P(x)y' + Q(x)y = 0 \quad \ldots\ldots \quad (2)$$

by dividing the leading coefficient $a_0(x)$.

A point x_0 is said to be an ordinary point of the differential equation (1) if both P(x) and Q(x) are analytic at x_0. A point that is not an ordinary point is said to be a singular point of the equation.

Assume that x_0 is an ordinary point (2), so that solutions in powers of $(x - x_0)$ actually do exists. Let

$$y = c_0 + c_1(x - x_0) + c_2(x - x_0)^2 + \ldots \quad (3)$$

be the solution of (2). Since the series (3) converges on an interval $|x - x_0| < R$ about x_0, it may be differentiated term by term on this interval twice in

succession to obtain

$$\frac{dy}{dx} = c_1 + 2c_2(x-x_0) + 3c_2(x-x_0)^2 + \ldots \quad (4)$$

$$\frac{d^2y}{dx^2} = 2c_2 + 6c_3(x-x_0) + 12c_4(x-x_0)^2 + \ldots \quad (5)$$

respectively. By substituting the series in the right members of (2), (3) and (4) for y and its first two derivatives ,respectively, in the differential equation (1), one can get the series solution. Simplifying the resulting expression it takes the following form

$$K_0 + K_1(x-x_0) + K_2(x-x_0)^2 + \ldots = 0 \quad (6)$$

Where the coefficients $K_i (i = 0,1,2,3\ldots)$ are functions of certain coefficients c_n of the solution (3).

By equating the coefficients to zero of each power of $(x-x_0)$ in the left member of (6) leads to a set of conditions that must be satisfied by the various coefficients c_n in the series (3). If the $c_n's$ are chosen to satisfy the set of conditions that thus occurs, then the resulting series (3) is the desired solution of the differential equation (1).

III. SAGE CODE FOR GETTING SERIES SOLUTIONS OF AN ODE THROUGH SYMBOLIC COMPUTATIONS

SAGE is an open-source mathematical software system that helps to perform many mathematical tasks such as plotting curves and surfaces, symbolic differentiation and integration and solving ODE etc[3]. SAGE have highly optimized functions that implement common numerical operations like integration, solving ordinary differential equations and solving systems of equations. It is almost a *viable, free, open source alternative to Magma, Maple, Mathematica, and Matlab.* Mathematician, scientist, or engineer can spend less time for doing tedious mathematical calculations by the way of using mathematical software and visualizing its solutions. Students can also benefit from mathematical software. The ability to plot functions and manipulate symbolic expressions easily can improve the understanding of mathematical concepts.

The following is the SAGE package sersol(p1x,p2x,p3x,p4x,x0,i1,i2,i3,N,nh,n) for obtaining series solution for third order ODE around the ordinary point **x** = x_0 and visualizing its solutions.

Input:

N (Number of terms of the series)

p1x, p2x , p3x, x_0, i1, i2, i3,n & nh

where p1x, p2x, p3x & p4x are the coefficients of y''', y'', y' & y respectively, i1, i2& i3 are initial conditions, n is highest order of the derivative and nh represents the non homogeneous term.

Output:

Series Solutions with visualization

```
def sersol(p1x,p2x,p3x,p4x,x0,i1,i2,i3,N,nh,n):
    x = var("x")
    z= var("z")
   cc=[0]*(N+2)
   dd=[0]*(N+1)
   ee=[0]*(N-2)
   aa = list(var('a_%d' % i) for i in (0..N))
   y = sum(a*(x-x0)**i for i,a in enumerate(aa))
   y1=y.substitute((x==z+x0))
   p1z=p1x.substitute((x==z+x0))
   p2z=p2x.substitute((x==z+x0))
   p3z=p3x.substitute((x==z+x0))
   p4z=p4x.substitute((x==z+x0))
   nh1=nh.substitute((x==z+x0))
  dy1=y1.derivative(z)
  d2y1=dy1.derivative(z)
  d3y1=d2y1.derivative(z)
  ode=p1z*d3y1+p2z*d2y1+p3z*dy1+p4z*y1-nh1;
  ode1=ode.simplify_full();
  ode2=ode1.collect(z)
  l2=len(ode2)
  bb=[0]*(l2+1)
  for k in range(1,N+1):
       m=z^k
       bb[k] = (ode2.coefficient(m)==0)
       ode2 = ode2 - ode2.coefficient(m)*m

  for k in range(N+1,l2):
       m=z^(k)
       ode2=ode2 - ode2.coefficient(m)*m
 bb[0]=(ode2==0);
 for k in range(0,N-(n-1)):
       dd[k] = bb[k]
 for k in range(0,N-(n-1)):
       ee[k] = aa[k+3]
 sol=solve(dd,(ee))
 c=sol[0];
 y2=0
for k in range(0,n):
     y2=y2+aa[k]*z^k
for k in range(0,N-(n-1)):

    y2=y2+(c[k].right_hand_side())*z**(k+n)

y3=y2.substitute(z=0)
y4=y2.derivative(z)
y5=y4.substitute(z=0)
y11=y4.derivative(z)
y12=y11.substitute(z=0)
y6=solve(y3==i1,a_0)
y7=solve(y5==i2,a_1)
y13=solve(y12==i3,a_2)
y8=y2.substitute(a_0=y6[0].right_hand_side())
y9=y8.substitute(a_1=y7[0].right_hand_side())
y14=y9.substitute(a_2=y13[0].right_hand_side())
y10=y14.substitute((z==x-x0))
return(y10)
```

Example 1

Consider the third order initial value problem given by: $y''' + \frac{1}{x}y' - \frac{1}{x^2}y = 0$ with $y(1) = 1, y'(1) = 0$ and $y''(1) = 1$

Output:

The series solution for N=8 is
$-(793/40320)(x-1)^8+(137/5040)(x-1)^7-(29/720)(x-1)^6+(1/15)(x-1)^5-(1/8)(x-1)^4+(1/6)(x-1)^3+(1/2)(x-1)^2+1$

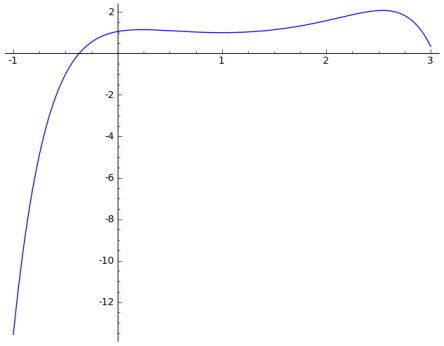


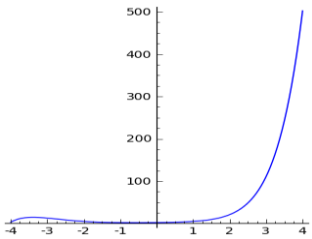
Figure 1: Visualizing solution and solution curve for the ordinary differential equation

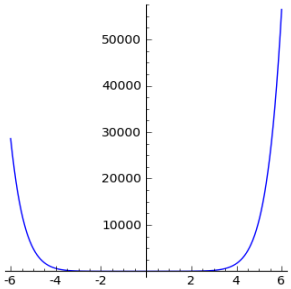
$y''' + \frac{1}{x}y' - \frac{1}{x^2}y = 0$ with
$y(1) = 1, y'(1) = 0, y''(1) = 1$ (N=8).

Example 2

Consider the following initial value problem
$y'' - xy' - y = 0, y(0) = 2, y'(0) = 1.$

Output:





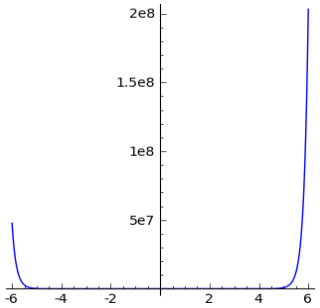


Figure 2: Visualizing solution curves for the ODE

$y'' - xy' - y = 0, y(0) = 2, y'(0) = 1$
for N=7,10 & 50

Example 3

Consider the following ordinary differential equation
$(x-1)y'' + y' + 2(x-1)y = 0, y(4) = 5, y'(4) = 0$
on the interval $4 \le x \le \infty$ around the ordinary point x_0=4

Output:

The series solution is
$(205/163296)(x-4)^8+(23/6804)(x-4)^7-(37/972)(x-4)^6-(2/27)(x-4)^5+(25/36)(x-4)^4+(5/9)(x-4)^3-5(x-4)^2+5$

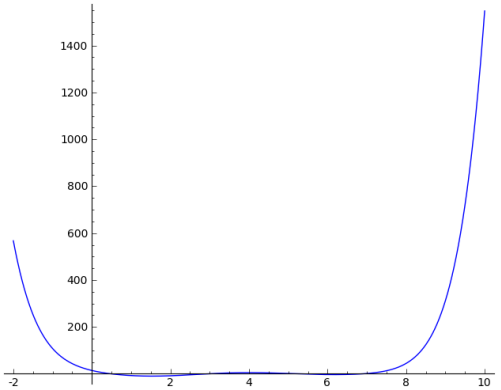


Figure 3: Visualizing solution and solution curve for the ODE

$(x-1)y'' + y' + 2(x-1)y = 0, y(4) = 5, y'(4) = 0$
and $4 \le x \le \infty$

Example 4

Comparison of series and exact solution curves for the following initial value problem

$y''' - 2y'' + y' = 2 - 24e^x + 40e^{5x}$,

$y(0) = \frac{1}{2}, y'(0) = \frac{5}{2}, y''(0) = -\frac{9}{2}$

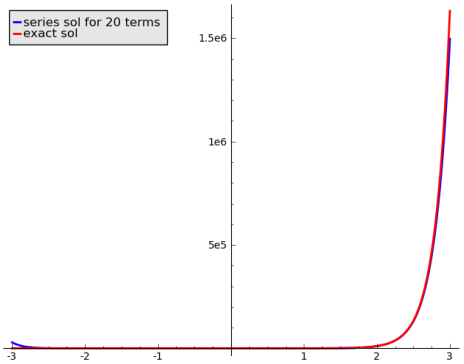


Figure 4: Comparison of series and exact solution of $y''' - 2y'' + y' = 2 - 24e^{x} + 40e^{5x}$, $y(0) = \frac{1}{2}, y'(0) = \frac{5}{2}, y''(0) = -\frac{9}{2}$

Example 5

Consider the following ordinary differential equation

$$y'' + xy' - y = e^{3x}$$

Output:

The series solution is

$(-1/13440)(5a_0-148)x^8+(1/240)(a_0+19)x^6$
$+(53/1680)x^7-(1/24)(a_0-8)x^4+(7/40)x^5$
$+(1/2)(a_0+1)x^2+(1/2)x^3+a_1x+a_0$

Where a_0 & a_1 are arbitrary constants

Example 6

Consider the following ordinary differential equation

$$y'' + \sin(x)y' + e^{x}y = 0$$

Output:

The series solution is

$(1/40320)(49a_0-110a_1)x^8-(1/5040)(37a_0+32a_1)x^7-$
$(1/720)(5a_0-14a_1)x^6+(1/20)(a_0+a_1)x^5+$
$(1/12)(a_0-a_1)x^4-(1/6)(a_0+2a_1)x^3-(1/2)a_0x^2+a_1x+a_0$

Where a_0 & a_1 are arbitrary constants

For the above cases, expand the trignometrical and exponential terms as a Taylor series expansion about origin upto the number of terms one require. Since there are no initial condition given, the solution will be in terms of arbitrary constants as given above.

The SAGE package is given for a third order equations and a simple modification in terms of the number of initial conditions and with the coefficients of derivative will give the solution of second order equations.

IV. LEGENDRE POLYNOMIALS

The Legendre differential equation $(1-x^2)y'' - 2xy' + n(n+1)y = 0 \ldots(7)$ where n=0,1,2,3,… is frequently encountered in physics and other technical fields. In particular, it occurs when solving Laplace's equation (and related partial differential equations) in spherical coordinates. The Legendre differential equation may be solved using the standard power series method. The equation (7) has regular singular points at $x = \pm 1$, its series solution about the origin will only converge for $|x| < 1$. When n is an integer, one of the series solution terminates depending on the nature of value of n. These solutions for $n = 0, 1, 2, 3, \ldots$ form a sequence of orthogonal polynomials called the Legendre polynomials.

The following SAGE Package legendre(p1x,p2x,p3x,N) solves the Legendre differential equation $(1-x^2)y'' - 2xy' + n(n+1)y = 0$ and computes set of Legendre polynomials with visualization.

Input:

N (Number of terms of the series)

p1x= $1-x^2$, p2x= $-2x$ & p3x=n^2+n

(Coefficients of y'', y' & y respectively)

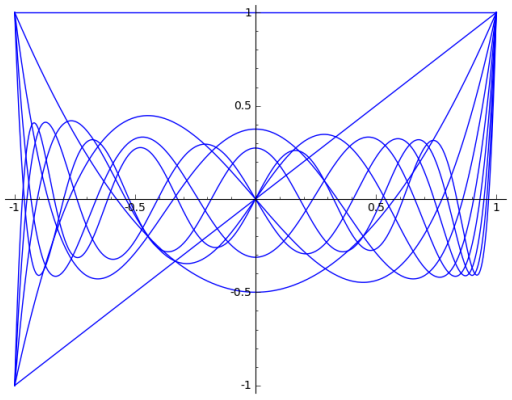
Output:

Legendre polynomials with visualization

```
def  legendre(p1x,p2x,p3x,N):
      x = var("x")
      n = var("n")
     N1= int(N/2)
     aa = list(var('a_%d' % i) for i in (0..N))
     bb = [0]*(N+2)
     cc = [0]*(N-1)
     dd = [0]*(N+1)
     ee = [0]*(N-1)
     ff = [0]*(N-1)
     gg = [0]*(N1)
     hh = [0]*(N1)
    y = sum(a*x**i for i,a in enumerate(aa))
    dy=y.derivative(x)
    d2y=dy.derivative(x)
    ode=p1x*d2y+p2x*dy+p3x*y;
    ode1=ode.simplify_full()
    ode2=ode1.collect(x)
    for k in range(1,N+1):
        m=x^k
        bb[k] = (ode2.coefficient(m)==0)
   ode2 = ode2 - ode2.coefficient(m)*m
   bb[0]=(ode2==0);
  for k in range(0,N-1):
      ee[k]=bb[k]
      cc[k]=aa[k+2]
  sol=solve(ee,(cc))
  d=sol[0]
  y2=aa[0]+aa[1]*x;
  for k in range(0,N-1):
      y2=y2+(d[k].right_hand_side())*x**(k+2)
  i=0
  for k in range(0,N-1 ,2):
```

```
        y3=y2.substitute(a_1=0,n=k)
        y4=y3.substitute(x=1)
        y5=solve(y4==1,a_0)
      gg[i]=y3.substitute(a_0=y5[0].right_hand_side())
      i=i+1
  i=0
  for k in range(1,N,2):
        y3=y2.substitute(a_0=0,n=k)
        y4=y3.substitute(x=1)
        y5=solve(y4==1,a_1)
      hh[i]=y3.substitute(a_1=y5[0].right_hand_side())
      i=i+1
  for k in range(0,N1):
        plot_list.append(plot(gg[k],(x,-1,1) ))
  for k in range(0,N1):
        plot_list.append(plot(hh[k],(x,-1,1) ))
  gr=sum(plot_list)
  gr.show()
  return(gg,hh)
```

Output:



([1, $(3/2)x^2$ - 1/2, $(35/8)x^4$ – $(15/4)x^2$ + 3/8, $(231/16)x^6$ – $(315/16)x^4$ + $(105/16)x^2$ - 5/16, $(6435/128)x^8$ – $(3003/32)x^6$ + $(3465/64)x^4$ – $(315/32)x^2$ + 35/128], [x, $(5/2)x^3$ – $(3/2)x$, $(63/8)x^5$ – $(35/4)x^3$ + $(15/8)x$, $(429/16)x^7$ - $(693/16)x^5$ + $(315/16)x^3$ – $(35/16)x$, $(12155/128)x^9$ – $(6435/32)x^7$ +$(9009/64)x^5$ – $(1155/32)x^3$ + $(315/128)x$])

Figure 3. Visualizing Legendre polynomials for N=10

V. HERMITE POLYNOMIALS

The Hermite differential equation $y'' - 2xy' + 2ny = 0$ is frequently encountered in physics and other technical fields. In particular, Hermite polynomials $H_n(x)$ arise in solving the Schrodinger equation for a harmonic oscillator. However, it also shows one way in which special functions arise from differential equations, so in that sense it is of interest to all.

The following SAGE Package hermitesol(p1x,p2x,p3x,N**)** solves the Legendre differential equation $y'' - 2xy' + 2ny = 0$ and computes set of Hermite polynomials for n=1,2,3… with visualization.

Input:
N (Number of terms of the series)
p1x=1,p2x = -2x & p3x = 2n
(Coefficients of $y'', y' \& y$ respectively)
Output:
Hermite polynomial with visualization

```
def hermitesol(p1x,p2x,p3x,N):
    x = var("x")
    n = var("n")
    N1=int(N/2)
    aa = list(var('a_%d' % i) for i in (0..N))
    bb = [0]*(N+2)
    cc = [0]*(N-1)
    dd = [0]*(N+1)
    ee = [0]*(N-1)
    ff = [0]*(N-1)
    gg = [0]*(N1)
    hh = [0]*(N1)
    y = sum(a*x**i for i,a in enumerate(aa))
    dy=y.derivative(x)
    d2y=dy.derivative(x)
    ode=p1x*d2y+p2x*dy+p3x*y;
    ode1=ode.simplify_full()
    ode2=ode1.collect(x)
    for k in range(1,N+1):
        m=x^k
        bb[k] = (ode2.coefficient(m)==0)
        ode2 = ode2 - ode2.coefficient(m)*m
    bb[0]=(ode2==0);

    for k in range(0,N-1):
        ee[k]=bb[k]
        cc[k]=aa[k+2]
    sol=solve(ee,(cc))
    d=sol[0]

    y2=aa[0]+aa[1]*x;
    for k in range(0,N-1):
        y2=y2+(d[k].right_hand_side())*x**(k+2)
  i=0
  for k in range(1,N,2):
      y3=y2.substitute(a_0=0,n=k)
      y4=y3.coefficient(x^k)
      y5=solve(y4==2**(k),a_1)
    gg[i]=y3.substitute(a_1=y5[0].right_hand_side())
    i=i+1

  i=1
  hh[0]=1;
  i=1
  for k in range(2,N-1,2):
      y3=y2.substitute(a_1=0,n=k)

      y4=y3.coefficient(x^k)

      y5=solve(y4==2**(k),a_0)

    hh[i]=y3.substitute(a_0=y5[0].right_hand_side())
    i=i+1
  plot_list=[]
```

```
    for k in range(0,N1):
      plot_list.append(plot(gg[k],(x,-2,2) ))

for k in range(0,N1):
     plot_list.append(plot(hh[k],(x,-2,2) ))

gr=sum(plot_list)
gr.show()
   return(gg,hh)
```

Output:

$[2x,8x^3-12x,32x^5-160x^3+120x,128x^7-1344x^5+3360x^3-1680x,\ 512x^9-9216x^7+48384x^5-80640x^3+30240x]$

$[1,4x^2-2,16x^4-48x^2+12,64x^6-480x^4+720x^2-120,\ 256x^8-3584x^6+13440x^4-13440x^2+1680]$

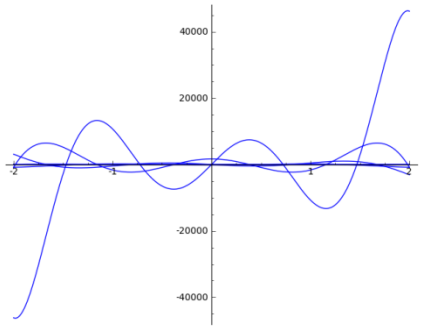


Figure 4: Visualizing Hermite polynomials for N=10.

VI. METHOD OF FROBENIOUS FOR REGULAR SINGULAR POINTS

If the linear equation $y''-P(x)y'+Q(x)y=0\cdots$ (8) has an irregular singularity at $x=x_0$ then the problem of finding series solution is not easy. If, however the above equation (8) has regular singular point at $x=x_0$ then one can develop a method for finding a series solutions, valid in neighborhood of x_0. This procedure is known as the method of Frobenius. In this article, the package for getting solutions valid in neighborhood of regular singular points ($x_0=0$) is developed, whose indicial equations are different and not differ by integer.

Input
N (Number of terms of the series)
p1x, p2x & p3x
(Coefficients of $y'', y' \& y$ respectively).

Output
Series solutions

```
def  frobenius(p1x,p2x,p3x,N):
      x = var("x")
      r = var("r")
      bb=[0]*(N+2)
      bb1=[0]*(N)
      cc=[0]*(N+1)
      dd=[0]*(N)
      y3=[0]*2*(N+1)
      aa = list(var('a_%d' % i) for i in (0..N))
      y = sum(a*x**i for i,a in enumerate(aa))
      Px=p2x/p1x;
      Qx=p3x/p1x;
       px=x*Px;
       p4x=px.simplify_full()
       qx=x^2*Qx;
       q4x=qx.simplify_full()
       a1=p4x.substitute(x=0)
       b1=q4x.substitute(x=0)
       sol=solve(r*(r-1)+a1*r+b1= =0,(r))

       for j in range(1,3):
          c=sol[j-1].right_hand_side()
          yp = y*x^(c)
          y1=yp.simplify_full();
          dy=y1.derivative(x)
          d2y=dy.derivative(x)
          ode1=p1x*d2y+p2x*dy+p3x*y1;
          ode2=ode1.simplify_full();
          ode=ode2/x^c;
          ode3=ode.collect(x)

          for k in range(0,N+1):
               m=x^k
             bb[k] = (ode3.coefficient(m)= =0)
             ode3 = ode3 - ode3.coefficient(m)*m
             bb[0]=(ode3= =0);

          for k in range(0,N):
               bb1[k]=bb[k]

          for k in range(1,N+1):
               dd[k-1]=aa[k]

          sol1=solve(bb1,(dd))
          sol2=sol1[0]
          y2=aa[0]*x^c
          for i in range(0,N-1):
          y2=y2+(sol2[i].right_hand_side())*x**(i+1+c)
          show(y2)
     return()
```

Example 7

Consider the following ordinary differential equation

$6x^2y''+7xy'-(1+x^2)y=0,(0<x<\infty)$

Since x = 0 is a regular singular points for the differential equation, output is shown below

The two linear independent solutions are

$(1/68078976)a_0x^{(15/2)}+(1/197904)a_0x^{(11/2)}+(1/1064)a_0x^{(7/2)}+(1/14)a_0x^{(3/2)}+(a_0/\sqrt{x})$

$(1/411374976)a_0x^{(25/3)}+(1/970224)a_0x^{(19/3)}+(1/3944)a_0x^{(13/3)}+(1/34)a_0x^{(7/3)}+a_0x^{(1/3)}$

Where a_0 is arbitrary constant

The above code can be modified for other singular points like one developed in this article and for other two cases of Frobenius such as the indicial equation's equal roots and roots differ by an integer for getting two independent solutions can be done in a similar way.

The results are verified for the example problems with [4], [5] and [6].

VII. COMPUTATION TIME COMPARISON

The Computation time comparison for obtaining the number of terms of the series for the ODE $y'' - xy' - y = 0, y(0) = 2, y'(0) = 1$ using SAGE (Wall time) and MATLAB (tic-toc time) are presented in Table 1.

Number of coefficients of the series	SAGE	MATLAB
50	1.252263212	3.712966
100	4.277512169	7.363384
150	10.05321922	11.65843
200	21.62953706	17.56204
250	32.45158839	24.66636
300	52.07608185	29.889488

Table 1: The Computation time comparison for obtaining the number of coefficients of the series for the ODE $y'' - xy' - y = 0, y(0) = 2, y'(0) = 1$ using SAGE and MATLAB

VIII. Conclusion

This computational tool will help the students to understand the nature of the solution and study the behavior of the solution through visualization. The figures confirm that the behavior of the solution changes considerably when the number of terms of the solution increases, which may not be possible without the use of Mathematical software. This is the first time a package has been developed through SAGE to visualize the series solution of any order. The programs will help not only solving the differential equations upto order three, Legendre and Hemite differential equations but also generating and visualizing solutions around ordinary, regular singular points, Legendre and Hermite polynomials for any order. One can extend the same concept for higher order differential equations of any order.

References

[1] Shepley L . Ross, Differential equations, Third edition, John Wiley & sons

[2] Erwin Kreyszig, Advanced Engineering Mathematics,nineth edition, John Wiley & Sons,Inc, 2006

[3] Craig Finch, Sage Beginner's Guide, Packt Publishing,2011

[4] Michael D.Greenberg, Advanced engineering Mathematics, Second edition, Printice Hall,1998

[5] Peter V O'neil, Advanced engineering Mathematics, International student edition, Thomson, 2007.

[6] Morris Tenenbaum, Harry Pollard, Ordinary Differential Equations, Dover publications, INC.,NewYork,1963