ISSN No:-2456-2165

Solving First Order Ordinary Differential Equations using Least Square Method: A comparative study

Parth Singh Pawar, Dhananjay R. Mishra, Pankaj Dumka*
Department of Mechanical Engineering, Jaypee University of Engineering and Technology, Guna-473226, India

Abstract:- In this research article an attempt has been made to examine the performance of Finite difference method (FDM) and Least square method (LSM) on the solution of first order ordinary differential equations (ODE). Both FDM and LSM are applied on the test problem and the results thus obtained are compared with the exact solution. It has been observed that for third degree basis function the results of LSM very close to the analytical result. On further increasing the degree the improvement in the result is very meagre and N=3 can be considered as the optimum solution for LSM. It has also been observed that the FDM is very sensitive to the number of grid points and deviates from the exact results by a substantive amount for lower number of nodes. Whereas the LSM is independent of the number of nodes.

Keywords:- *Least Square Method; Finite Difference Method; Ordinary Differential Equation; Python; Optimization.*

Nomenclature:

x	independent variable
y	dependent variable
i	degree of basis function
j	Node index
D	differential operator
w_i	weight
R	residue
N	Number of weights
E	squared residuals
y_{exact}	exact solution
$\tilde{y}$	approximate function
ϕ_i	basis function
Δx	spacing between nodes
LSM	Least square Method
FDM	Finite difference method
ADM	Adomian Decomposition Method
HPM	Homotopy Perturbation Method

I. INTRODUCTION

At the very core of every physical phenomenon lies some sort of mathematical relation in the form of the differential equation [1]. The behaviour of these equations can be linear or non-linear [2]. So, several researchers have devised many methods to solve these differential equations [3]. Till very recent the solution of differential equations is mostly dominated by numerical computations but, very recently the analytical methods have gained popularity [4].

For solving differential equations semi-analytically Adomian Decomposition Method (ADM) has been adopted by researchers [5], [6]. For non-linear equations Homotopy Perturbation Method (HPM) and Variational Iteration Methods (VIM) are good methods [7]. Perturbation method is another method but due to its drawback it is not that frequently adopted for the solution of ODE's [8]. The problems associated with linear stability in solving differential equations have been talked by Hajmohammadi and Nourazar [9].

Weighted residual-based methods are the schemes which are approximation techiniques which are also adopted to solve differential equations. Ozisik first introduced Least square method (LSM) and Galerkin techiniques which are based on weighted residuals [10]. The solution of third order differential equation based on collection method has been introduced by Stern and Rasmussen [11].

Finite difference method (FDM) is a very old method to solve differential equations [12]. This is based on Taylor series expansion [13]. In this method domain is divided into nodes and the differential equation is discretized at each node.

In this research article, LSM based solution of first order linear differential equation has been reported. FDM has also been applied on the problem and the results of both the methods are compared with the exact analytical solution. This will dictate the accuracy of the LSM over FDM.

II. PROBLEM STATEMENT

In in this research article, we will be focusing on following one dimensional ordinary differential equation (ODE):

$$\frac{dy}{dx} - y = 0 \qquad (1)$$

where, $0 \leq x \leq 1$.

As this is a first order ODE so this will have only one boundary condition. Let say that when $x = 0$ the value of $y = 1$(viz. $y(0) = 1$). The exact solution of the Eq. 1 is e^x which will act as a benchmark to our optimization solution.

III. LEAST SQUARE METHOD (LSM)

Least square method is one of the methods which is based on weighted residuals minimization. In this method a trial function is introduced in the parent differential equation and then the residue is minimized. Let us consider a boundary value problem as follows:

$$D(y) - f(x) = 0 \qquad (2)$$

where, *D* is the differential operator.

To start the LSM, it is assumed that the dependent variable y is estimated by an approximation function $\tilde{y}$ which is composed of coefficients/weights (w_i) and basis function (ϕ_i) [14]. The basis functions are picked from linearly independent set of functions in the projection space. The $\tilde{y}$ can be written as:

$$\tilde{y} = \sum_{i=1}^{N} w_i \phi_i \quad (3)$$

Where *i* varies from 1 to n. Now the target is to obtain w_i by least square mechanism as follows:

Let y_{exact} be the exact solution of the differential Eq. 2 i.e., y_{exact} once replaced in Eq. 2 will result in zero (as shown in Eq. 4).

$$D(y_{exact}) - f(x) = 0 \quad (4)$$

Whereas if we replace $\tilde{y}$ in the Eq. 2 the result will not be zero as this is not the exact solution. This non-zero value which the Eq. 2 return when approximate solution is plugged-in into it is what we call as Residue (*R*) which can be written as:

$$R(x, \tilde{y}) = D(\tilde{y}) - f(x) \neq 0 \quad (5)$$

Now the concept of LSM is to make the residue tend to zero by minimizing the error function in D^2 norm, so that the weight coefficients can be evaluated as follows:

$$E = \int_x R^2(x, \tilde{y}) dx \quad (6)$$

The optimum solution is obtained once E is set to minimum viz.:

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \int_x R^2(x, \tilde{y}) dx = 0 \quad (7)$$

As *i* varies from 1 to *N*, so Eq. 7 will result in N linear equations which can be solved for N unknown i.e., $w_1, w_2 \ldots w_N$. Proper choosing of ϕ_i is very essential in LSM so for the problem in hand we will go for a polynomial function. The choice should be such that the approximate solution should satisfy the boundary conditions.

➢ *LSM applied to the problem*

Applying the algorithm discussed in section 3 onto the Eq. 1 will result into following 5 steps:

- As the problem is of first order so choosing a polynomial basis function for guess solution.

$$\tilde{y} = \sum_{i=1}^{N} w_i x^i + y_0 \quad (8)$$

- As it was already mentioned that the approximate solution should satisfy the boundary condition so the boundary condition $y(0) = 1$ can only be satisfied if $y_0 = 1$.

- Developing expression for residue by plugging $\tilde{y}$ from Eq. 8 into Eq. 1.

$$R(x, \tilde{y}) = \frac{d}{dx}(\tilde{y}) - \tilde{y} = \frac{d}{dx}\left(\sum_{i=1}^{N} w_i x^i + y_0\right) - \left(\sum_{i=1}^{N} w_i x^i + y_0\right) \quad (9)$$

- Minimizing the square error

$$\frac{\partial E}{\partial w_i} = 2 \int_{x=0}^{x=1} R(x) \frac{\partial R(x)}{\partial w_i} dx = 0, i = 1, \ldots, N \quad (10)$$

- Now solving linear equation for different values of N and comparing it with the exact solution.

In this research article Python has been used to solve the problem symbolically. N is varied from 1 to 5 to check the results for different simulations. Appendix section shows the Python code developed for the evaluation of different weights and data plotting.

The approximate function thus obtained for different N's are enumerated in Table 1.

Table 1: Approximate solutions for different N.

N	Approximate solution
1	$\tilde{y} = \frac{3}{2}x + 1$
2	$\tilde{y} = \frac{70}{83}x^2 + \frac{72}{83}x + 1$
3	$\tilde{y} = \frac{2485}{8884}x^3 + \frac{945}{2221}x^2 + \frac{2250}{2221}x + 1$
4	$\tilde{y} = \frac{126126}{1810709}x^4 + \frac{254240}{1810709}x^3 + \frac{921942}{1810709}x^2 + \frac{1809000}{1810709}x + 1$
5	$\tilde{y} = \frac{4178559}{300698723}x^5 + \frac{10498950}{300698723}x^4 + \frac{51177210}{300698723}x^3 + \frac{150115840}{300698723}x^2 + \frac{601429185}{601397446}x + 1$

IV. FINITE DIFFERENCE METHOD (FDM) AND ITS APPLICATION

In case of finite difference, the domain is divided into n number of discrete points. The governing equation is discretised, and the discretised equation is evaluated at each grid point. In case of FDM the Taylor series expansion is used to evaluate the derivatives. As the problem in hand is first order ODE so backward difference is used to discretize the governing equation.

Let $f(x)$ be the function whose derivative is to be evaluated at some j^{th} location then the Taylor series can be written as:

$$f(x - \Delta x) = f(x) - \frac{df}{dx}\Delta x + \frac{d^2 f}{dx^2}\frac{\Delta x^2}{2!} - \frac{d^3 f}{dx^3}\frac{\Delta x^3}{3!} + \cdots \quad (11)$$

where, Δx is the distance between 2 grid points. Now the first derivative can be written as:

$$\frac{df}{dx} = \frac{f(x)-f(x-\Delta x)}{\Delta x} + O(\Delta x) \quad (12)$$

Above equation is the backward difference approximation of first order. Also, this approximation is first order accurate.

➢ *FDM applied to the problem*

Applying the above-mentioned method to the Eq. 1 we will get:

$$\frac{dy}{dx} - y = \frac{y(x)-y(x-\Delta x)}{\Delta x} - y(x) = 0 \quad (13)$$

When it comes to computation the x will correspond to the j^{th} node and $(x - \Delta x)$ corresponds to the node just before it viz. $(j - 1)$. Now Eq. 13 then becomes:

$$\frac{y_j - y_{j-1}}{\Delta x} - y_j = 0 \quad (14)$$

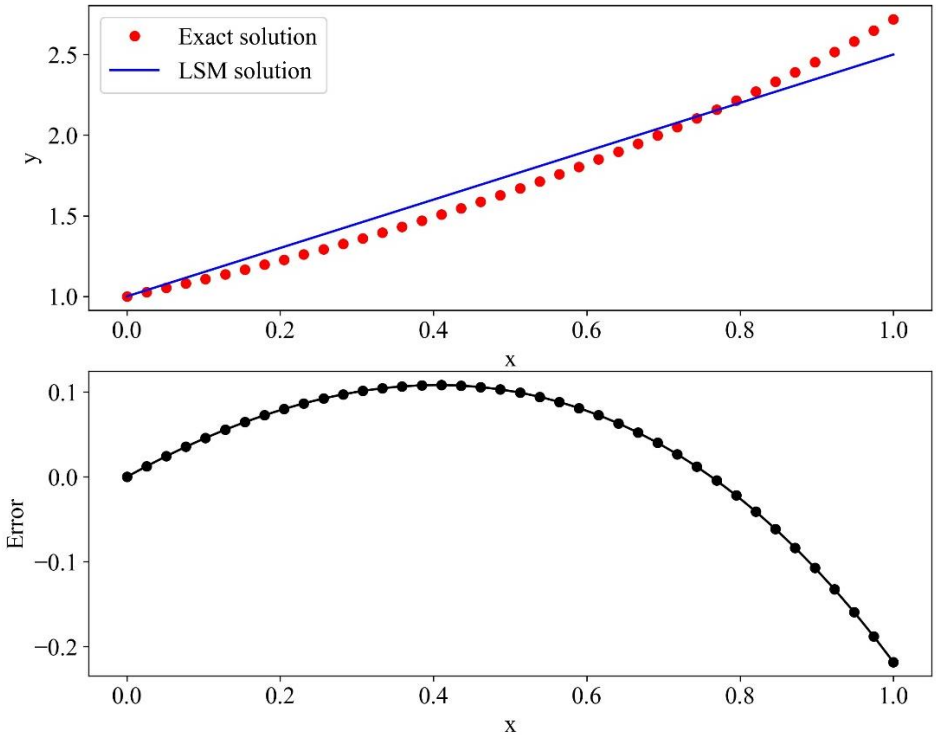
On further simplification Eq. 14 becomes:

$$y_j = y_{j-1}/(1 - \Delta x) \quad (15)$$

Now Eq. 15 is the finite difference discretization of Eq. 1 which is be solved for each node in the domain.

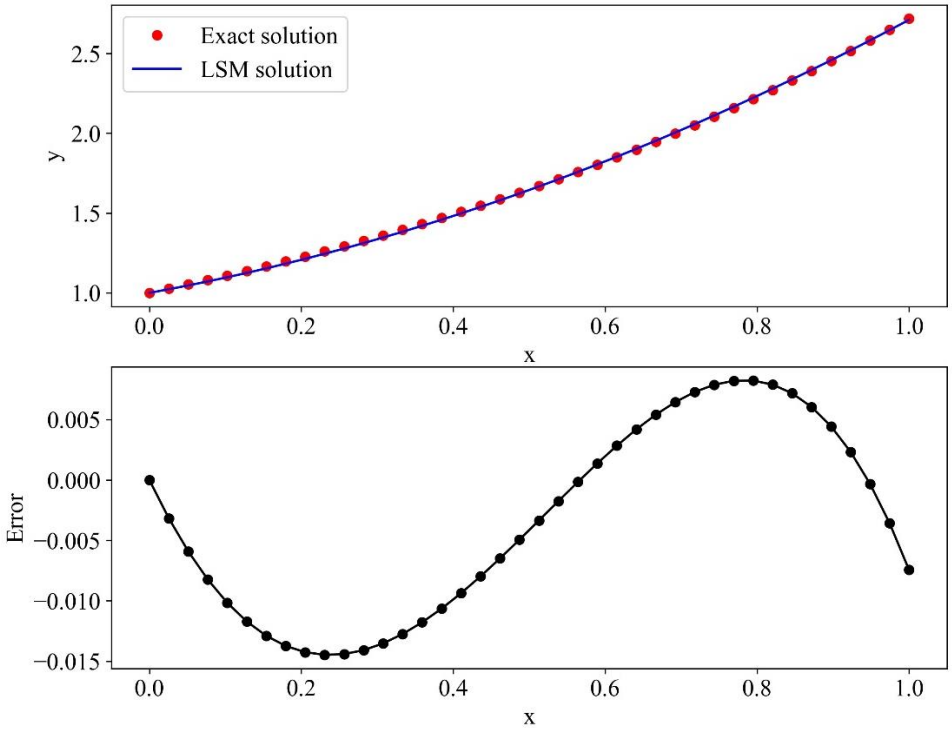
V. RESULTS AND DISCUSSION

The objective of this study is to evaluate the performance of LSM to evaluate the solution of first order ODE and its comparison with the Euler's method. Figure 1(a) shows the variation of LSM output, exact solution, and error as a function of domain length for different values of N. For N=1 the LSM returns a liner relation between x and y which is not what the exact solution tells hence, large deviation and error. For N=2 (Figure 1(b)) the LSM results are close to exact and still there is a little bit of error. When N is increased to 3 (Figure 1(c)), one can see that the error has reduced a lot in comparison to N=1 and 2. And for N=4 and 5 (Figure 1(d) and 1(e)) the error has almost reached zero.

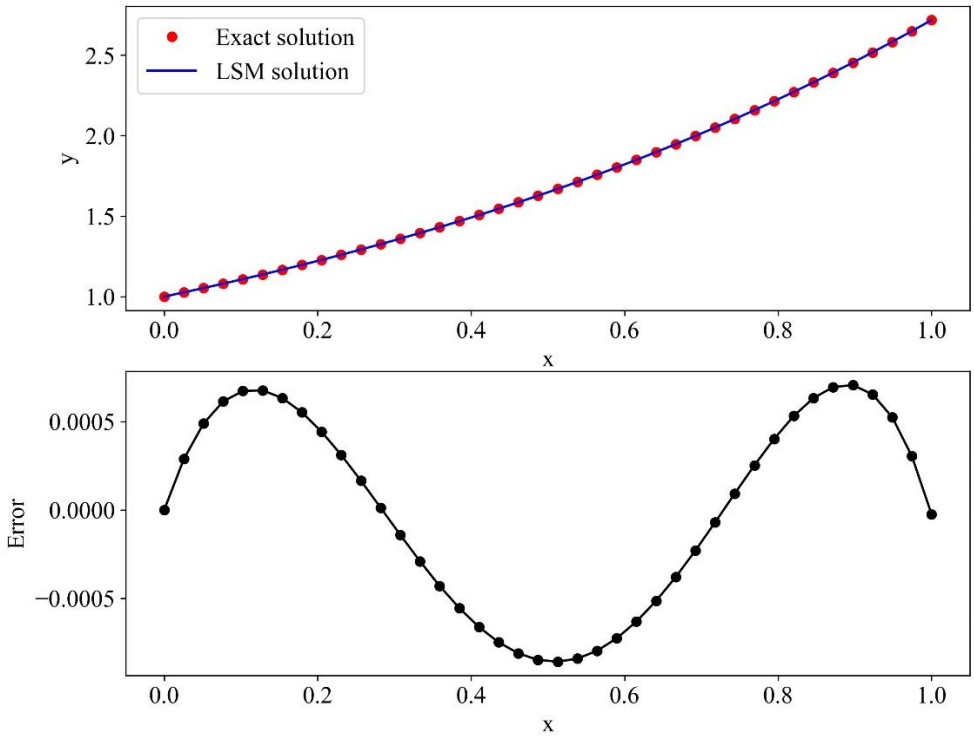


(a) N=1

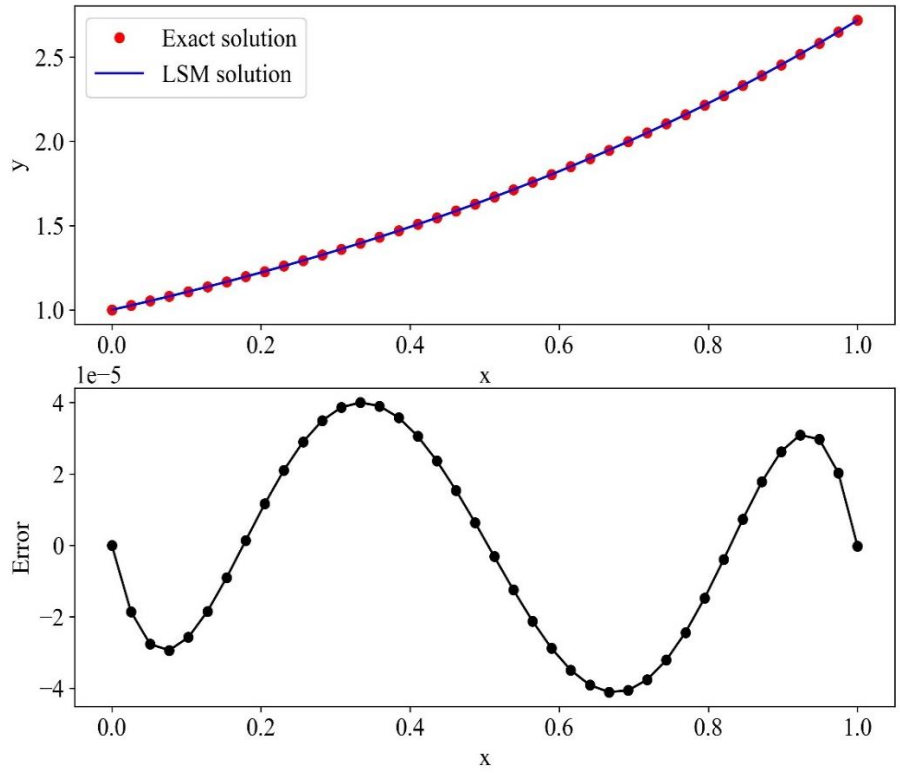
ISSN No:-2456-2165



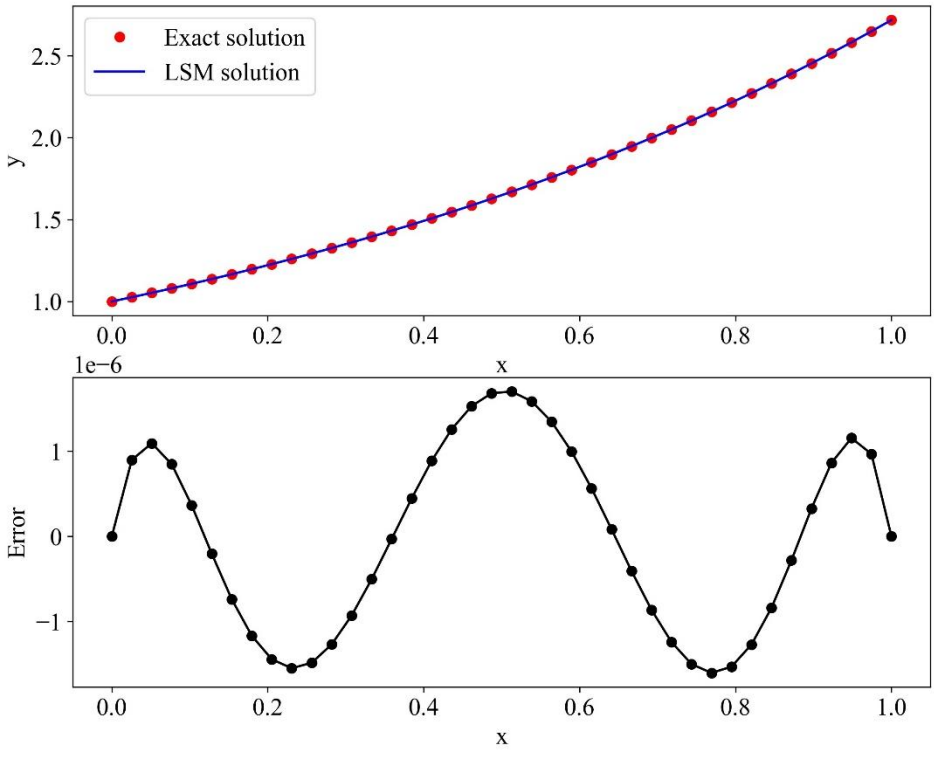
(b) N=2



(c) N=3



(d) N=4



(e) N=5

Fig 1: Variation of y and error as a function of x for different N

One more thing to observe from Figure 1 is that as the order of basis function increases the lobes in the error function also increase. The error presented is the exact error hence once can se its variation both in positive and negative direction. The number of lobes in the error plot is representative of the degree of polynomial. For N=1 the polynomial is one so there is only one lobe and at last for N=5 as the polynomial is of 5^{th} order so the number of lobes in error plot is five, likewise for other values of N.

ISSN No:-2456-2165

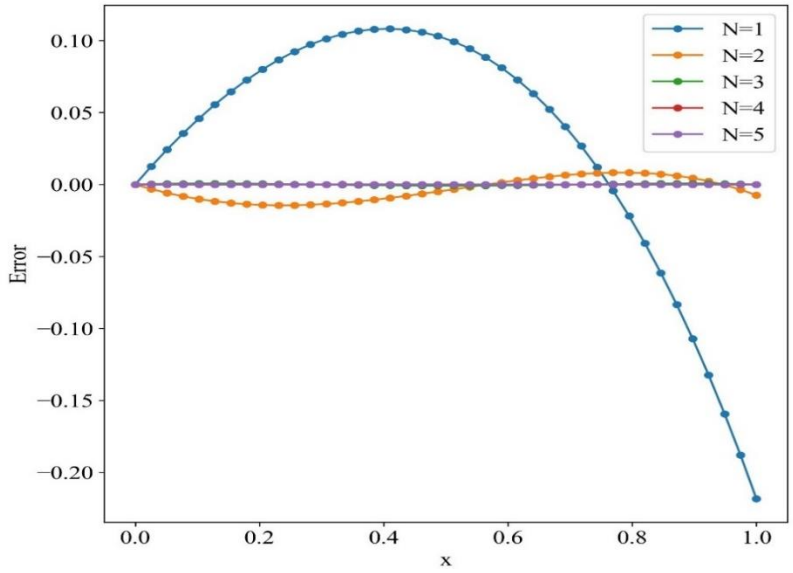


Fig 2:- Error variation for N varying from 1 to 5 as a function of x

Figure 2 depicts the error for all N's in one figure. As the N goes from 1 to 3 the error the error reduces drastically to a very small value. In fact, N=3 can be considered as the optimum solution of the problem as beyond it the degree of approximating function output improves to a higher value with only a meagre change in the function output and error. To further investigate into the problem Figure 3 show the variation of N from 3 to 5. Here one can see that for N=4 and 5 the error is almost zero but with extra cost of computation time. Hence, N=3 is the optimum degree of approximating function as after it the error reduces at diminishing rate.

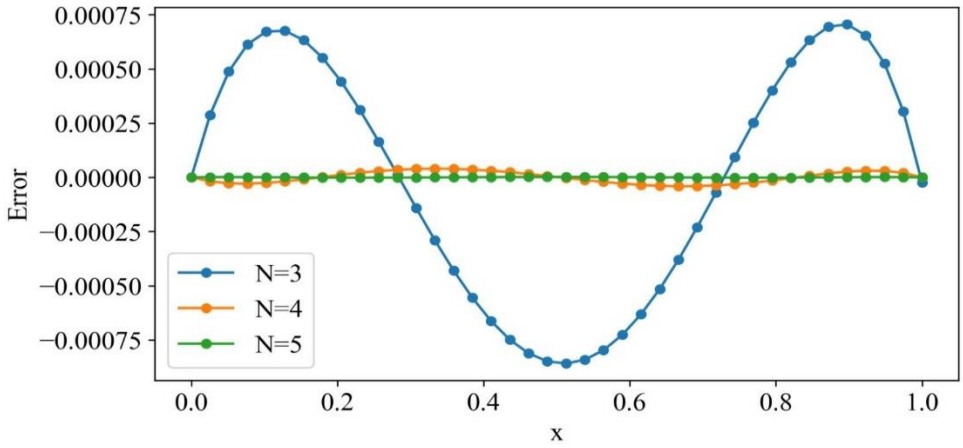


Fig 3: Error variation for N varying from 3 to 5

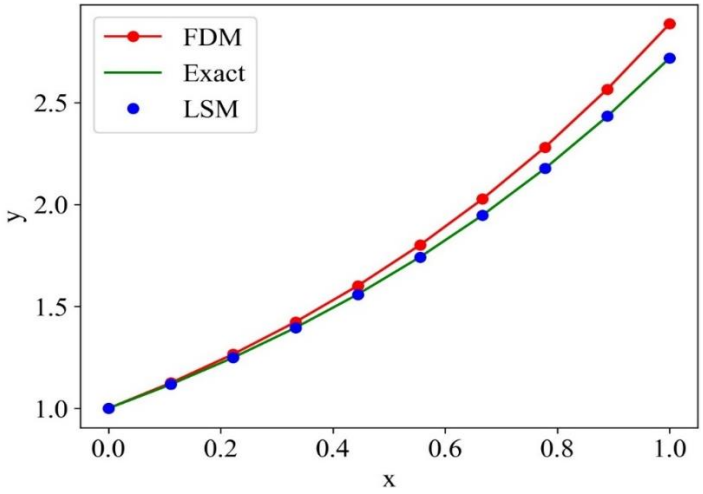


Fig 4: Variation of output variable from FDM solution, LSM solution, and exact solutions as a function of x

Figure 4 show a comparison between solution obtained from LSM & FDM and their comparison with the exact solution. The domain is divided into 10 equal parts and one can see that there is a substantive variation between the result predicted from FDM and LSM. The accuracy of FDM is dependent on how division of the domain is done, more the divisions more will be the accuracy. Whereas the LSM is independent of the number of nodes and it gives more accurate result even for fewer number of nodes.

VI. CONCLUSIONS

Based on the LSM and FDM results following conclusions can be drawn:

- Computation wise LSM is faster than FDM.
- Polynomial can be used as a basis funciton for first order ODE's.
- With increase in the order of polynomial the accuracy of the solution is improved.
- After N=3 the solution improves at diminishing rate hence, at N=3 one can get the optimum solution of first order ODE at low cost of computation.
- FDM is very sensitive to the number of nodes. Higher the number of nodes more the accuracy of FDM. Increasing the number of nodes will result in more computation time and cost.
- LSM is independent of the number of nodes and hence can give better accuracy in comparison to FDM.

REFERENCES

[1] R. W. Easton, *Ordinary Differential Equations: An Introduction to Nonlinear Analysis (Herbert Amann)*, vol. 33, no. 4. Walter de gruyter, 1991.
[2] S. Wang, H. Wang, and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed DeepONets," *Sci. Adv.*, vol. 7, no. 40, 2021.
[3] G. E. Latta and G. M. Murphy, *Ordinary Differential Equations and Their Solutions.*, vol. 68, no. 4. Courier Corporation, 1961.
[4] P. Kunkel and V. Mehrmann, *Differential-algebraic equations: analysis and numerical solution*. European Mathematical Society, 2006.
[5] D. J. Evans and K. R. Raslan, "The Adomian decomposition method for solving delay differential equation," *Int. J. Comput. Math.*, vol. 82, no. 1, pp. 49–54, 2005.
[6] J. Biazar, E. Babolian, and R. Islam, "Solution of the system of ordinary differential equations by Adomian decomposition method," *Appl. Math. Comput.*, vol. 147, no. 3, pp. 713–719, 2004.
[7] J. H. He, "Variational iteration method for autonomous ordinary differential systems," *Appl. Math. Comput.*, vol. 114, no. 2–3, pp. 115–123, 2000.
[8] A. A. Hemeda, "Homotopy perturbation method for solving partial differential equations of fractional order," *Int. J. Math. Anal.*, vol. 6, no. 49–52, pp. 2431–2448, 2012.
[9] M. R. Hajmohammadi and S. S. Nourazar, "On the solution of characteristic value problems arising in linear stability analysis; semi analytical approach," *Appl. Math. Comput.*, vol. 239, pp. 126–132, 2014.
[10] M. N. Ozisik, *Boundary Value Problems of Heat Conduction (Dover Phoenix Editions) (Dover Phoneix Editions)*. courier Corporation, 2002.
[11] R. H. Stern and H. Rasmussen, "Left ventricular ejection: Model solution by collocation, an approximate analytical method," *Comput. Biol. Med.*, vol. 26, no. 3, pp. 255–261, 1996.
[12] M. Necati Özişik, H. R. B. Orlande, M. J. Colaço, and R. M. Cotta, *Finite difference methods in heat transfer: Second Edition*. CRC press, 2017.
[13] Y. Ren, B. Zhang, and H. Qiao, "A simple Taylor-series expansion method for a class of second kind integral equations," *J. Comput. Appl. Math.*, vol. 110, no. 1, pp. 15–24, 1999.
[14] B. Hashemi and Y. Nakatsukasa, "Least-squares spectral methods for ODE eigenvalue problems," 2021.

APPENDIX

A. Python code for the evaluation of weights

```
from sympy import *
c,s,n,x,c1,c2,c3,c4,c5,y_g=symbols('c,s,n,x,c1,c2,c3,c4,c5,y_g')

c=[c1,c2,c3,c4,c5]

n=int(input('Enter the value of N upto which you want to solve'))

#Creating Basis function
s=1
for i in range(1,n+1):
   s=s+c[i-1]*x**i
y_g=s

#Residue creation
symbols('R')
R=diff(y_g,x)-y_g

# Minimizing the square error
E=integrate(R**2,(x,0,1))
EE=[]

for i in range(1,n+1):
   EE.append(diff(E,c[i-1]))

Eqns=(EE)
a=solve(Eqns,c)

y_g.subs(a)
```

B. Python code for data plotting

- *y vs x and error vs x for different cases*

```
from matplotlib.pylab import *
font = {'family' : 'Times New Roman',
        'size'   : 16}
matplotlib.rc('font', **font)
x=linspace(0,1,40)
y_exact=exp(x)

#N=1
#y=3*x/2+1

#N=2
#y=70*x**2/83+72*x/83+1

#N=3
#y=2485*x**3/8884+945*x**2/2221+2250*x/2221+1

#N=4
#y=126126*x**4/1810709+254240*x**3/1810709+921942
*x**2/1810709+\
1809000*x/1810709+1

#N=5
y=4178559*x**5/300698723+10498950*x**4/300698723+
\
51177210*x**3/300698723+150115840*x**2/300698723+
\
601429185*x/601397446+1

# Error evaluation
error=(y-y_exact)

figure(1,dpi=300)
tight_layout()
# plot 1:
plt.subplot(2, 1, 1)
plot(x,y_exact,'ro',label='Exact solution')
plot(x,y,'b-',label='LSM solution')
xlabel('x')
ylabel('y')
legend()

# plot 2:
plt.subplot(2, 1, 2)
plot(x,error,'k-o')
xlabel('x')
ylabel('Error')
savefig('N=5.jpg')
show()
```

- Single error plot for all the cases

```
from matplotlib.pylab import *
font = {'family' : 'Times New Roman',
        'size'   : 16}
matplotlib.rc('font', **font)
x=linspace(0,1,40)
y_exact=exp(x)

y1=3*x/2+1

y2=70*x**2/83+72*x/83+1

y3=2485*x**3/8884+945*x**2/2221+2250*x/2221+1

y4=126126*x**4/1810709+254240*x**3/1810709+921942
*x**2/1810709+\
1809000*x/1810709+1

y5=4178559*x**5/300698723+10498950*x**4/300698723
+\
51177210*x**3/300698723+150115840*x**2/300698723+
\
601429185*x/601397446+1

y=[y1,y2,y3,y4,y5]

# Error evaluation
error=(y-y_exact)


figure(1,dpi=300)
for i in range(2,len(error)):
   plot(x,error[i],'-o',label=f'N={i+1}',markersize=3)


xlabel('x')
ylabel('Error')
legend()
savefig('Error for 3 to 5.jpg')
show()
```

C. FDM code and comparison plot

```
from pylab import *
font = {'family' : 'Times New Roman',
        'size'   : 16}
matplotlib.rc('font', **font)
n=10
x=linspace(0,1,n)
Δx=1/(n-1)
y=zeros(n)

y[0]=1

for i in range(1,n):
   y[i]=y[i-1]/(1-Δx)

figure(1,dpi=300)
plot(x,y,'r-o',label='FDM')

plot(x,exp(x),'g-',label='Exact')

y_LSM=2485*x**3/8884+945*x**2/2221+2250*x/2221+1
plot(x,y_LSM,'bo',label='LSM')
xlabel('x')
ylabel('y')
legend()
savefig('FDM_LSM.jpg')
show()
```