South East Asian J. of Mathematics and Mathematical Sciences
Vol. 20, No. 1 (2024), pp. 137-154

DOI: 10.56827/SEAJMMS.2024.2001.11 **ISSN (Online): 2582-0850**

ISSN (Print): 0972-7752

SERIES SOLUTION OF ORDINARY DIFFERENTIAL EQUATION USING A MODIFIED VERSION OF THE ADOMIAN DECOMPOSITION METHOD

Mithun Bairagi

Mangal Chandi High School, Khosalpur,
Patrasayer, Bankura - 722206, West Bengal, INDIA

E-mail : bairagirasulpur@gmail.com

(**Received:** *May 04, 2023* **Accepted:** *Apr. 22, 2024* **Published:** *Apr. 30, 2024)*

Abstract: We present a new modified version of the Adomian decomposition method for computing the series solutions of the nonlinear ordinary differential equations (ODEs). The recently proposed Adomian matrix algorithm is used in this method to compute the Adomian polynomials for scalar-valued nonlinear polynomial functions, which allows us to get the series solution of the ODEs numerically and makes it much faster than symbolic computation. This method can test the convergence of the series solution of the ODE by calculating the global squared residual error of the solution. Several types of nonlinear ODEs, such as Abel equation, De Boer-Ludford equation, Van der Pol equation, Painleve-Ince equation, and Falkner-Skan equation, are solved using this method to illustrate its performance and effectiveness in delivering solutions.

Keywords and Phrases: ODE, series solution, convergence, Adomian polynomials.

2020 Mathematics Subject Classification: 34-XX, 34-04, 34A25, 40A05.

1. Introduction

Nonlinear differential equations are effective modeling tools for nonlinear dynamical events in a variety of disciplines, including mathematical biology, nonlinear optics, plasma physics, nano physics, solid-state physics, and fluid dynamics. Finding exact solutions to nonlinear differential equations can be accomplished using a number of well-known techniques, for example, F-expansion method [15, 44], first integral method [26] Lie symmetry method [18], Laplace–Fourier transform technique [36], and so on. Unfortunately, analytical techniques cannot be used to tackle the majority of nonlinear situations. In addition, to solve nonlinear problems, standard numerical methods require perturbation, discretization, linearization, or transformation. It is important to note that there are a number of semianalytical approaches in the literature for solving nonlinear differential equations, including the Adomian decomposition method (ADM) [5, 6, 7, 8, 37], the homotopy analysis method (HAM) [31], and the homotopy perturbation method (HPM) [10]. For nonlinear systems, these methods essentially construct a series solution iteratively where we must solve a linear differential equation at each iteration. The Adomian decomposition method has been widely used in the literature to get series solution of a differential equation, [1, 2, 8, 13, 20, 21, 22, 33]. This technique can provide an analytical approximation to the exact solutions in the series form that converge very rapidly [1, 2, 20]. In this method, no linearization or perturbation is required. The Laplace Adomian decomposition method (LADM), which is created by combining the Adomian decomposition method and the Laplace transform, is a powerful method. As can be observed in [11, 12, 29, 30, 38], LADM has also been extensively utilized to discover the series solutions of fractional-order nonlinear differential equations.

In the aforementioned works [1, 2, 8, 13, 20, 21, 22, 33], ADM has been applied to differential equations for series solutions, and notably, there need some symbolic computations to get the series solutions, which slows the calculating process. Series solutions to ODEs can also be determined using a popular technique called automatic differentiation (AD) [17, 27, 28] that computes derivatives of the Taylor coefficients numerically with respect to the point of expansion without truncation errors, which is very fast and much more efficient than using symbolic algebra. AD is a set of techniques based on the mechanical application of the well-known chain rule to obtain derivatives of any arbitrary function at a given point specified by a computer program. AD manipulates the fact that every computer program, no matter how complicated, performs a sequence of elementary arithmetic operations such as addition, subtraction, multiplication, division, and elementary functions such as $\exp, \log, \sin$, and $\cos$. By repeatedly applying the chain rule of derivative calculus to these operations, derivatives of arbitrary orders can be calculated automatically and accurately to working precision.

However, in this paper, we have used a new modified version of the Adomian decomposition method for computing the series solutions of the nonlinear ordinary differential equations (ODEs) numerically by employing the one-dimensional case of the recently proposed Adomian matrix algorithm [16] for fastest computations of Adomian polynomials for scalar-valued nonlinear polynomial functions. The Adomian matrix algorithm determines Adomian polynomials of scalar-valued nonlinear polynomial functions with the help of matrix formulations rather than recurrence processes, and does not require complex mathematical operations such as parametrization, expansion, regrouping, and differentiation. Therefore, we need not compute the differentiations using the well-known chain rule of a composite function like AD, which speeds up the solution process more efficiently. This modified version of the Adomian decomposition method can determine the series solution of the nonlinear ODE (2) which should be polynomial about x (independent variable),u (dependent variable), and its derivatives, and free of any parameters. Following the method described in [23, 39], we can also test the convergence rate of the obtained series solution. This modified version of ADM approach has been implemented in an open-source Python package called *odeSolu* [1]. The primary computing routine (determination of Adomian polynomials) in this package is written in Cython, which increases its computing efficiency. In this package, following [23, 39] we have also implemented a method to test the convergence rate of the obtained series solution.

The paper is organized as follows: In Sec. 2 we describe the basic concepts of the Adomian Decomposition Method. In Sec. 3, we present the modified version of ADM to get the series solution of an ODE with a convergence test. In Subsec. 3A, we apply the modified version of ADM to some ODEs using the Python package *odeSolu*, and compare the calculated series solution with the numerical solutions of the ODEs using SciPy's module *scipy.integrate.odeint*. In Sec. 4, we discuss our results and make some conclusions on our works.

2. Adomian Decomposition Method

In this section we discuss the key concepts of the Adomian Decomposition Method following [24, 34, 35]. Let us consider a nonlinear ODE in order p with independent variable x (real and scalar) and dependent variable u having the general form [34, 35]

$$\mathcal{N}u = g(x), \tag{1}$$

where $\mathcal{N} : H \to H$ is the nonlinear operator from a Hilbert space H into H. In ADM, it is assumed that $\mathcal{N}$ to be decomposed into

$$Lu + Ru + Nu = g(x), \tag{2}$$

where $L : H \to H$ is the highest-order linear differential operator $L[.] = \frac{d^p}{dx^p}[.]$ which

[1]`https://github.com/mithun218/odeSolu.git`

is assumed to be invertible, $R : H \to H$ is a linear differential operator containing the linear derivatives of less order than L, $N : H \to H$ is an analytic nonlinear operator containing all other nonlinear terms, $g(x) \in H$ is given analytic function. Here we should note that choosing the operator L is not always unique [9, 40, 41]. It is also noteworthy that in Eq. (2), u is a scalar function of the real variable x. u will be a vector-valued function for a set of differential equations. However, in this paper, our studies are limited to single ODE in which u is a scalar-valued function. The principle step of the decomposition method is to assume a series solution

$$u = \sum_{i=0}^{\infty} u_i, \tag{3}$$

and then, the ADM scheme that corresponds to the functional equation (2) converges strongly to $u \in H$, which is the unique solution to the functional equation [19, 34]. By the Eq. (3), the nonlinear term Nu is decomposed into an infinite series

$$Nu = \sum_{i=0}^{\infty} A_i, \tag{4}$$

where A_i are the well-known Adomian polynomials which depend on the solution components $u_0, u_1, \ldots, u_i$. The following definitional formula introduced by G. Adomian [5-7, 21], determines the Adomian polynomials for a given nonlinear functional $Nu = F(u)$ ($F(u)$ is assumed to be an analytic functional in Hilbert space H):

$$A_M = \frac{1}{M!} \frac{d^M}{d\lambda^M} F\left(\sum_{i=0}^{\infty} u_i \lambda^i\right)\Bigg|_{\lambda=0}, \quad M = 0, 1, 2, \ldots, \tag{5}$$

where the analytic parameter λ is simply a grouping parameter. One crucial characteristic of the Adomian polynomial A_M is that it is constructed solely to depend on the solution components $(u_0, u_1, \ldots, u_M)$ and not on higher-order solution components u_i with $i > M$, as mentioned by Wazwaz and Azreg [14, 42]. As a result, the higher-order terms for $i > M$ do not contribute to the summation in Eq. (5). Using Eqs. (2), (3) and (4), the solution components u_M can be determined using the following classic Adomian recursion scheme

$$u_0 = \Phi + L^{-1}[g(x)], \tag{6a}$$
$$u_{M+1} = -L^{-1}[R[u_M]] - L^{-1}[A_M], \quad M = 0, 1, 2, \ldots, \tag{6b}$$

where Φ is determined by the initial conditions such that $L[\Phi] = 0$. However, in practice, it is impossible to calculate the series solution up to infinite terms. But, one can get the approximate solution very close to the exact solution by including a large number of terms in Eq. (3). In this context, it is notable that there are several alternate recursion schemes [1, 4, 25, 43], which differ from one another by the other choice of the initial solution component u_0 for computational convenience. There are number of works on the convergence of the Adomian series solutions obtained from the various recursion schemes [1, 2, 3, 20, 23, 32, 39].

3. Modified version of ADM

This section aims to present a new modified version of ADM for the fastest determinations of the series solution up to higher-order terms. We derive the power series solution of the ODE (2) with initial conditions using the classic Adomian recursion scheme (6) with the help of the one-dimensional case of the Adomian matrix algorithm [16]. Using the Adomian matrix algorithm, we have introduced a modified version of ADM by which one can determine the series solution of the ODE numerically besides some preliminary symbolic computations at the initial stage, which makes this method faster. This modified version of ADM is described in the following steps:

i. Let us consider an ODE in order p, with independent variable x and dependent variable u in the general form (2) of which the series solution has to be determined. At first, in Eq. (2), the single highest derivative term is isolated on the left-hand side (L.H.S.), and the remaining terms are moved to the right-hand side (R.H.S.). After these isolation processes, Eq. (2) becomes

$$Lu = g(x) - Ru - Nu. \tag{7}$$

Equation (7) can also be written in the following form

$$u_{(p)} = f(x, u_{(1)}, u_{(2)}, \dots, u_{(p)}), \tag{8}$$

where $Lu = u_{(p)}$. The number within the first bracket in the subscript of u denotes differentiation with respect to x, e.g., the pth derivatives of u with respect to x is represented by $u_{(p)}$. Here it is worth mentioning that this method is only applicable when the R.H.S. in Eq. (8) is polynomial about x, u, and its derivatives, and also it is free of any parameters. These conditions allow us to express Eq. (8) in the following summation form

$$u_{(p)} = \sum_{i=0}^{q}\sum_{j=0}^{q}\sum_{k_1=0}^{q}\sum_{k_2=0}^{q}\dots\sum_{k_p=0}^{q} C_{ijk_1k_2\dots k_p} x^i u^j u_{(1)}^{k_1} u_{(2)}^{k_2} \dots u_{(p)}^{k_p}, \tag{9}$$

where $C_{ijk_1k_2\ldots k_p}$ are real numerical constant values. The ODE (8) is in the sum of product forms in the summation (9) where $p+2$ is the number of terms in each product and q is the number of product terms in the summation.

ii. In order to avoid any symbolic computations in later stages, now we convert Eq. (9) into a matrix whose elements are real numbers, and it bears all information about the R.H.S. of the ODE (7). We can write Eq. (9) in $q \times (p+2)$ matrices

$$D = \begin{pmatrix} C_{0000\ldots0} & 0 & 0 & 0 & 0 & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots & \ldots & \vdots \\ C_{13450\ldots0} & 1 & 3 & 4 & 5 & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots & \ldots & \vdots \\ C_{qq\ldots q} & q & q & q & q & \ldots & \ldots & \ldots & q \end{pmatrix}. \tag{10}$$

Each row of the matrix (10) having $p+2$ number of elements represents each product term in the summation (9). All the elements in the 1st column represent the constant coefficients of the product terms. The elements in 2nd, 3rd, 4th, 5th,... $(p+2)$th columns represent the powers of $x, u, u_{(1)}, u_{(2)}, \ldots, u_{(p)}$ respectively. We can divide the matrix into two parts, one represents the linear part of ODE $(g(x) - Ru)$, and another represents the nonlinear part $(-Nu)$ of the ODE (7). We only require to calculate the Adomian matrix for the nonlinear part. In such a way, the whole information of the R.H.S. of the ODE (9) are stored in the matrix (10), and this matrix is the numerical representation of the R.H.S. of the ODE (7).

iii. Let us consider we have to determine n number of terms in the power series solution. Then, the solution can be expressed in the following power series form

$$u(x) = \sum_{i=0}^{n-1} a_i x^i. \tag{11}$$

We assume p number of initial conditions as

$$u(0) = c_0, u_{(1)}(0) = c_1, u_{(2)}(0) = c_2, \ldots, u_{(p-1)}(0) = c_{p-1}, \tag{12}$$

where $c_i (i = 0, 1, \ldots p-1)$ are real numbers. If $g(x)$ is the polynomial function with degree $r-1$ in independent variable x, we can write

$$g(x) = \sum_{i=0}^{r-1} g_i x^i, \tag{13}$$

where $g_i (i = 0, 1, \ldots r-1)$ are real numbers. Then using Eq. (13), we can calculate $L^{-1}g(x)$ which is given by

$$L^{-1}g(x) = \int \ldots \int g(x) d^p x = \sum_{i=0}^{r-1} \frac{g_i i!}{(i+p)!} x^{i+p}. \tag{14}$$

Using the initial conditions (12) and Eq. (14), from Eq. (6a) we get

$$u_0 = \sum_{i=0}^{p-1} \frac{c_i}{i!} x^i + \sum_{i=0}^{r-1} \frac{g_i i!}{(i+p)!} x^{i+p} = \sum_{i=0}^{s-1} d_i x^i, \tag{15}$$

where

$$\left.\begin{aligned} & s = p, \\ d_i &= \frac{c_i}{i!} + \frac{g_i i!}{(i+p)!}, \text{ for } i = 0, 1, \ldots, r-1, \\ &= \frac{c_i}{i!}, \text{ for } i = r, r+1, \ldots, s-1, \end{aligned}\right\} \text{if } p > r, \tag{16}$$

and

$$\left.\begin{aligned} & s = r, \\ d_i &= \frac{c_i}{i!} + \frac{g_i i!}{(i+p)!}, \text{ for } i = 0, 1, \ldots, p-1, \\ &= \frac{g_i i!}{(i+p)!}, \text{ for } i = p, p+1, \ldots, s-1. \end{aligned}\right\} \text{if } p < r, \tag{17}$$

and

$$\left.\begin{aligned} & s = p = r, \\ d_i &= \frac{c_i}{i!} + \frac{g_i i!}{(i+p)!}, \text{ for } i = 0, 1, \ldots, s-1, \end{aligned}\right\} \text{if } p = r. \tag{18}$$

Using Eq. (15), Eq. (11) can be written as

$$u(x) = \sum_{i=0}^{n-s} u_i = \sum_{i=0}^{s-1} d_i x^i + \sum_{i=s}^{n-1} a_i x^i. \tag{19}$$

In Eq. (19), $a_i (i = s, s+1, \ldots, n-1)$ are unknowns, have to be determined in the later steps using u_0 given in (15) and Eq. (6b). Notably, if $d_j = 0$ for $p-1 < j < s-1$ with $s > p$ in Eq. (19), then $a_j (p-1 < j < s-1)$ are also unknowns which will be calculated in similar way like a_i. Now, we express Eq. (15) in terms of a $1 \times n$ row matrix

$$S = \begin{pmatrix} d_0 & d_1 & d_2 & \ldots & d_{s-1} & 0 & 0 & \ldots & 0 \end{pmatrix}. \tag{20}$$

S is called the solution matrix. The first s elements in S are the real numbers, which are the constant coefficients of x^i in the first part of Eq. (19). One can note that the remaining $(n-s)$ elements of S are filled with zero. In the following step, the values of $a_i(i = s, s+1, \ldots, n-1)$ are determined using the recursion scheme (6b) and the last $(n-s)$ elements of S are filled with their values. Here, we should note that if $d_j = 0$ for $p-1 < j < s-1$ with $s > p$ in Eq. (20), we also have to determine the unknowns $a_j(p-1 < j < s-1)$ in the similar way.

iv. This is the crucial and challenging step among all the steps in the method. This step elapses most of the computation time in solving an ODE. In order to determine the unknowns $a_j(p-1 < j < s-1), a_i(i = s, s+1, \ldots, n-1)$ in Eq. (19), we use the solution matrix S and apply the recursion scheme (6b). We start the calculations of the values of the unknowns from the leftmost (i.e. the lowest column) position of the matrix S. At first, to compute the values of the unknowns $a_j(p-1 < j < s-1)$, we have to extract the information of each term in the ODE through its matrix representation (10), and perform some mathematical operations on S. The mathematical operations are multiplications by a scalar, element-wise matrices multiplications, addition, derivatives, integration, and of course, the determinations of Adomian polynomials of the nonlinear terms in the ODE. Adomian polynomials are determined following the Adomian matrix algorithm the one-dimensional case [16]. To calculate the derivative and integration of a one-dimensional matrix, we use the algorithms described in Listings 1 and 2 respectively. The function *doDerivM* in Listing 1 written in pseudo-code is the implementation of a simple procedure to determine the derivative of a one-dimensional matrix numerically. On the other hand, in Listing 2, the function *doIntegM* implements an easy method to perform integration on a one-dimensional matrix numerically. After calculating a_j, the solution matrix S is updated with the element a_j in the j-th column and S becomes

$$S = \begin{pmatrix} d_0 & d_1 & d_2 & \ldots & a_j & \ldots & d_{s-1} & 0 & \ldots & 0 \end{pmatrix}. \tag{21}$$

Then, by repeatedly applying the above procedure of this step-iv, one can easily compute all other unknowns a_i for $i = s, s+1, \ldots, n-1$, and finally, we get the power series solution of Eq. (2) in matrix S.

Listing 1 The algorithm in pseudo-code for finding derivatives of a one-dimensional matrix A.

```
input: One-dimensional matrix $A$
output: $p$-times differentiation of $A$
function doDerivM(S, p)
  Construct matrix $C$ which contains whole numbers
  $\{0, 1, 2, \ldots, n-1\}$: C $\leftarrow \begin{pmatrix} 0 & 1 & 2 & \ldots & n-1 \end{pmatrix}$
  for i $\leftarrow$ 1 to $p$ do
    Element-wise multiplication: $T \leftarrow \begin{pmatrix} 0 & T_1 & T_2 & \ldots & T_{n-1} \end{pmatrix} \leftarrow C \circ A$
    Shift the elements of $T$ leftward, then the first element 0 will
        become the last element and the remaining will shift left:
    R $\leftarrow \begin{pmatrix} T_1 & T_2 & \ldots & T_{n-1} & 0 \end{pmatrix}$
  end for
  return R
end function
```

Listing 2 The algorithm in pseudo-code for finding integrations of a one-dimensional matrix A.

```
input: One-dimensional matrix $A$
output: $p$-times integration of $A$
function doIntegM(S, p)
  Construct matrix $C$ which contains the number sequence
  $\{1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{n-1}, 0\}$: C $\leftarrow \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \ldots & \frac{1}{n-1} & 0 \end{pmatrix}$
  for i $\leftarrow$ 1 to $p$ do
    Element-wise multiplication: $T \leftarrow \begin{pmatrix} T_1 & T_2 & \ldots & T_{n-1} & 0 \end{pmatrix} \leftarrow C \circ A$
    Shift the elements of $T$ rightward, then the last element 0 will
        become the first element and the remaining will shift
        right:
    R $\leftarrow \begin{pmatrix} 0 & T_1 & T_2 & \ldots & T_{n-1} \end{pmatrix}$
  end for
  return R
end function
```

Convergence test of the solution: To test the convergence rate of the Adomian series solution (19) obtained from this modified version of ADM, we follow [23, 39]. According to [23, 39], the global error associated with the series solution (for sufficiently large truncation level M) over the interval $\Omega = [a, b]$

$$u = \sum_{i=0}^{M} u_i, \tag{22}$$

(above equation (22) is obtained from (19) at stage M) defined by [39]

$$\mathcal{R}(x) = Lu + Ru + Nu - g(x). \tag{23}$$

If Eq. (23) is square integrable over the domain Ω, then the global squared residual error (Res) is defined by

$$Res = \int_a^b \mathcal{R}^2(x)dx. \tag{24}$$

If one get smaller value of Res that is tending to zero, then the solution u_M has a good convergence rate within the interval $\Omega = [a, b]$, and it is a good approximation to the solution of (2). In this context, one can note that in [39], authors have used modified recursion scheme where an convergence parameter h is inserted in the classical Adomian recursion scheme (6) for speeding up its convergence. In modified recursion scheme we have to determine the best possible value of h by minimizing the global squared residual error for getting the fastest convergence rate of the ADM series, requiring some symbolic computations that slow down the calculating process of solutions of Eq. (2). However, in the present work we are interested in rapid computer-generation of the Adomian series solutions of Eq. (2) to higher orders numerically, and for this purpose, we have used the classical Adomian recursion scheme (6).

3A. Series solutions of some ODEs

We have developed a Python package *odeSolu* [2], which implements the modified version of ADM (described in the above Sec. 3) for getting a power series solution of an ODE. In this subsection, we present some examples of ODEs that have been solved using *odeSolu* to demonstrate the efficiency and effectiveness of the modified version of ADM. We have selected various kinds of nonlinear ODE, including higher nonlinearity and higher-derivative terms, they are listed below:

(1) The Abel equation of the first kind

$$\frac{du}{dx} + 0.2x^2u^3 + 0.1xu^2 + 5u + 4 = 0, \tag{25}$$

with the initial condition $u(0) = 1$.

(2) A second-order ODE with quartic nonlinearity

$$\frac{d^2u}{dx^2} + 0.1\frac{du}{dx} + u^4 + 4 = 0, \tag{26}$$

subject to the initial conditions $u(0) = 0, u_1(0) = 1$.

[2]`https://github.com/mithun218/odeSolu.git`

(3) The De Boer-Ludford equation

$$\frac{d^2u}{dx^2} - u^4 + x^2u = 0, \qquad u(0) = 1, u_1(0) = 1. \tag{27}$$

(4) The Van der Pol equation

$$\frac{d^2u}{dx^2} - 0.05\left(1 - u^2\right)\frac{du}{dx} + u = 0, \qquad u(0) = 0, u_1(0) = 0.5. \tag{28}$$

(5) The Painleve-Ince equation

$$\frac{d^2u}{dx^2} + 3u\frac{du}{dx} + u^3 = 0, \qquad u(0) = 0, u_1(0) = 0.5. \tag{29}$$

(6) A Falkner–Skan equation

$$\frac{d^3u}{dx^3} + u\frac{d^2u}{dx^2} - 2\left(\frac{du}{dx}\right)^2 + 2 = 0, \qquad u(0) = 1, u_1(0) = 0.5, u_2(0) = 1. \tag{30}$$

(7) A fourth-order ODE

$$\begin{aligned}&\frac{d^4u}{dx^4} - x^2\frac{d^3u}{dx^3} + 3xu\frac{d^2u}{dx^2} - 6\frac{du}{dx} + 2x^2 + x = 0,\\ &u(0) = 0, u_1(0) = 0.5, u_2(0) = 1, u_3(0) = 1.\end{aligned} \tag{31}$$

(8) A fifth-order ODE with quintic nonlinearity

$$\begin{aligned}&\frac{d^5u}{dx^5} - 0.001u^2\frac{d^4u}{dx^4} - 2xu\left(\frac{d^3u}{dx^3}\right)^2 + 0.5xu\left(\frac{d^2u}{dx^2}\right)^4 - \frac{du}{dx} + x^2u^3 = 0,\\ &u(0) = 1, u_1(0) = 0, u_2(0) = 1, u_3(0) = 1, u_4(0) = 0.5.\end{aligned} \tag{32}$$

We have solved all the above-listed ODEs using *odeSolu* package, and the calculated series solutions are shown in Figs. 1 and 2. The red and orange dotted lines in all plots display the series solutions for different values of n (the number of series terms). The blue dotted lines express the numerical solutions of the ODEs obtained using SciPy's module *scipy.integrate.odeint*, which uses the LSODA algorithm to solve ODEs. From the plots, we observe that all the series solutions satisfy the numerical solutions very well within the convergence limits of the power series.

The *odeSolu* package is very fast in calculating the series solutions. However, the elapsed time increases with the degree of nonlinearity and the value of n. For example, it takes almost 25 second in calculating the series solution of Eq. (32) containing fifth-order nonlinearity for $n = 500$.

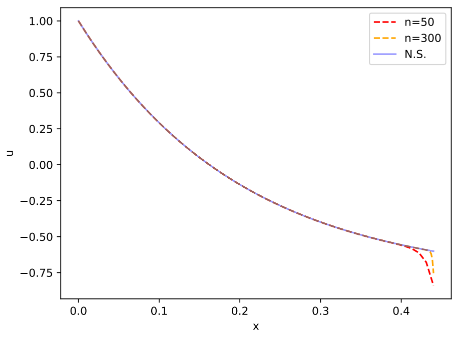
Table 1: Global squared residual error (Res) for Eqs. (25)-(32) in the intervals $\Omega = [a, b]$ within the convergence limits. CPU times (in second) of computing the solutions are given within the parenthesis in second column.

ODE	Number of series terms n	Ω	Res
(25)	50(0.19)	[0, 0.42]	0.0369
	300(1.65)	[0, 0.42]	6.762e-12
(26)	50(0.18)	[0, 1.0]	1.999
	500(9.71)	[0, 1.0]	1.110e-21
(27)	50(0.22)	[0, 1.36]	0.3647
	500(8.96)	[0, 1.36]	8.487e-29
(28)	50(0.16)	[0, 3.55]	29.456
	500(5.12)	[0, 3.55]	1.479e-3
(29)	50(0.13)	[0, 1.92]	41.794
	500(4.9)	[0, 1.92]	4.159e-12
(30)	50(0.09)	[0, 2.25]	26766.32
	500(0.46)	[0, 2.25]	1.220e-16
(31)	100(0.16)	[0, 2.0]	3.697e-8
	1000(1.32)	[0, 2.0]	2.267e-26
(32)	50(0.44)	[0, 1.4]	0.572
	500(24.66)	[0, 1.4]	1.259e-28

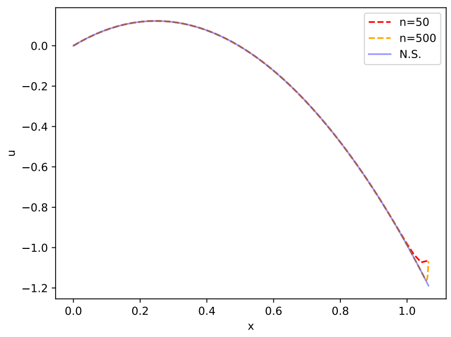
We have also calculated the global squared residual error (Res) defined in Eq. (24), and listed the estimated values of Res for different values of series terms n in Table 1. It is clear from Table 1 that the values of Res are very close to zero increasing n in a given interval $\Omega = [a, b]$ within the convergence limit. Therefore, the estimated series solutions become good approximation to the solutions of Eqs. (25)-(32) by including larger series terms in the convergence limits $\Omega = [a, b]$.

4. Conclusion

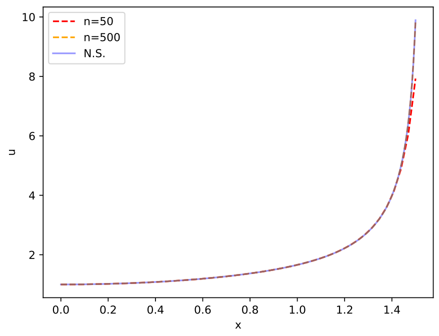
We have developed a new modified version of the Adomian decomposition method (ADM) to get the series solutions of the ODEs by using the one-dimensional case of the recently proposed Adomian matrix algorithm [16].



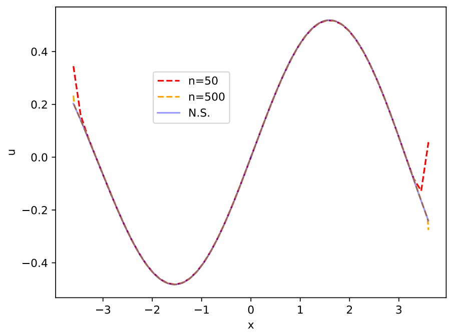
(a)



(b)



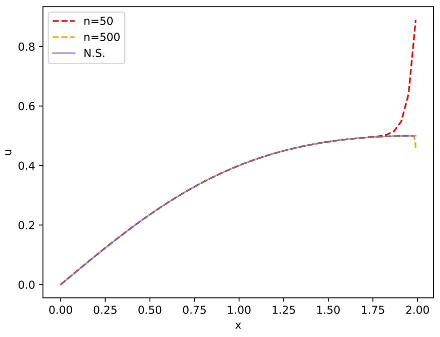
(c)



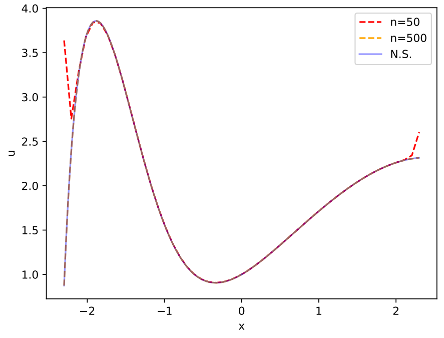
(d)

Figure 1: The power series solutions are compared with numerical solutions (N.S.). The red and orange dotted lines represent the series solutions of Eqs. (25), (26), (27), (28), in Figs. 1a, 1b, 1c, 1d, respectively. The blue dotted lines represent the numerical solutions.

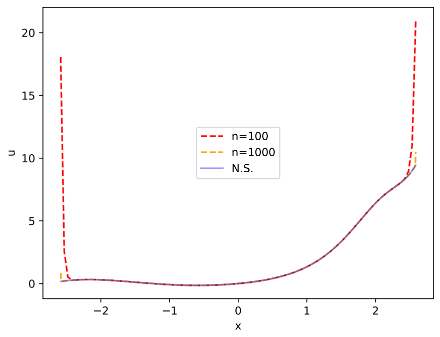
This method determines the series solutions of the ODEs numerically using the classic Adomian recursion scheme (6), which is very fast. This method can test the convergence of the computed series solution by estimating the global squared residual error (*Res*) defined in Eq. (24). We have developed an open-source Python package called *odeSolu* that implements this modified version of ADM. We have applied the package to various types of ODEs that include higher nonlinearity and higher-derivative terms, and compared the series solutions with numerical solutions in Figs. 1 and 2. From Figs. 1 and 2, we have observed that the series solutions are very close to the numerical solutions within the convergence limits. Using *odeSolu*, we have also calculated the parameter *Res* of the series solutions of these ODEs, as depicted in Table 1. From this Table 1, we have observed that when we include many series terms in the series solutions, the values of Res tend to zero, and the series solutions have a good convergence rate in the given intervals $\Omega = [a, b]$ within the convergence limits, and these solutions are good approximations to the solutions of these ODEs. Our proposed methods are very fast and efficient. However, it spends more time to give the solution when the degree of nonlinearity of the ODE and the number of terms n are increased. It is important to note that this modified method works only in the case where the R.H.S. in Eq. (8) is polynomial about x, u and its derivatives, and also it is free of any parameters.



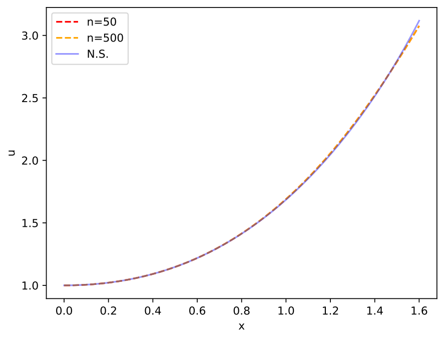
(a)



(b)



(c)



(d)

Figure 2: The power series solutions are compared with numerical solutions (N.S.). The red and orange dotted lines represent the series solutions of Eqs. (29), (30), (31), (32), in Figs. 2a, 2b, 2c, 2d, respectively. The blue dotted lines represent the numerical solutions.

In future studies, we can extend the use of this modified ADM approach to the analysis of the system of ODEs and partial differential equations (PDEs).

References

[1] Abbaoui K. and Cherruault Y., Convergence of Adomian's method applied to differential equations, Comput. Math. Appl., 28 (1994), 103–109.

[2] Abbaoui K. and Cherruault Y., New ideas for proving convergence of decomposition methods, Comput. Math. Appl., 29 (1995), 103–108.

[3] Abdelrazec A. and Pelinovsky D., Convergence of the Adomian decomposition method for initial-value problems, Numer. Methods Partial Differential Equations, 27 (2011), 749–766.

[4] Adomian G. and Rach R., Noise terms in decomposition solution series, Comput. Math. Appl., 24 (1992), 61–64.

[5] Adomian G., Stochastic System, Academic Press, New York, 1983.

[6] Adomian G., Nonlinear Stochastic Operator Equations, Academic Press, Orlando, 1986.

[7] Adomian G., Nonlinear Stochastic Systems Theory and Applications to Physics, Kluwer Academic, Dordrecht, 1989.

[8] Adomian G., Solving Frontier Problems of Physics: The Decomposition Method, Kluwer Academic, Dordrecht, 1994.

[9] Adomian G., Rach R. and Shawagfeh N. T., On the analytic solution of the Lane–Emden equation, Found. Phys. Lett., 8 (1995), 161–181.

[10] Agirseven D. and Ozis T., An analytical study for Fisher type equations by using homotopy perturbation method, Computers and Mathematics with Applications, 60 (2010), 602-609.

[11] Ali A., Humaira, Laila and Shah K., Analytical solution of General Fisher's Equation by using Laplace Adomian decomposition method, J. Pure Appl. Math, 2 (2018), 01.

[12] Ali A., Shah K. and Khan R. A., Numerical treatment for traveling wave solutions of fractional Whitham-Broer-Kaup equations, Alexandria Engineering Journal, 57 (2018), 1991.

[13] Alkresheh H. A., New classes of Adomian polynomials for the Adomian decomposition method, International Journal of Engineering Science Invention, 5 (2016), 37.

[14] Azreg-Aï nou M., A developed new algorithm for evaluating Adomian polynomials, Comput. Model. Eng. Sci., 42 (2009), 1–18.

[15] Bairagi M., GiNaCDE: the high-performance F-expansion and First Integral Methods with C++ library for solving Nonlinear Differential Equations, Journal of Open Research Software, 7 (2022) 3885.

[16] Bairagi M., A New Algorithm to Determine Adomian Polynomials for Nonlinear Polynomial Functions, Asian Research Journal of Mathematics, 19 (2023), 1-12.

[17] Bendtsen C. and Stauning O., FADBAD, a flexible C++ package for automatic differentiation, Department of Mathematical Modelling, Technical University of Denmark, (1996).

[18] Bluman G. W. and Anco S. C., Symmetry and Integration Methods for Differential Equations, Springer, New York, 2002.

[19] Bougoffa L., J. Appl. Math. Comput., 43 (2013) 31–54.

[20] Cherruault Y., Convergence of Adomian's method, Kybernetes, 18 (1989), 31–38.

[21] Duan J. S., New recurrence algorithms for the nonclassic Adomian polynomials, Computers and Mathematics with Applications, 62 (2011), 2961.

[22] Duan J. S., Rach R. and Wazwaz A. M., A reliable algorithm for positive solutions of nonlinear boundary value problems by the multistage Adomian decomposition method, Open Engineering, 5 (2014), 59–74.

[23] Duan J. S. and Rach R., A new modification of the Adomian decomposition method for solving boundary value problems for higher order nonlinear differential equations, Applied Mathematics and Computation, 218 (2011), 4090-4118.

[24] Duan J. S., An efficient algorithm for the multivariable Adomian polynomials, Appl. Math. Comput., 217 (2010), 2456–2467.

[25] Duan J. S., Recurrence triangle for Adomian polynomials, Appl. Math. Comput., 216 (2010), 1235–1241.

[26] Feng Z., On Explicit Exact Solutions to the Compound Burgers-KdV Equation, Physics Letters A., 293 (2002), 57-66.

[27] Griewank A., Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, Penn., 2000.

[28] Griewank A., Juedes D. and Utke J., Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. ACM Trans. Math. Software, 22 (1996), 131-167.

[29] Haq F., et al., Numerical Solution of Fractional Order Epidemic Model of a Vector Born Disease by Laplace Adomian Decomposition Method, Punjab University Journal of Mathematics, 49 (2017), 13.

[30] Haq F., Shah K., Rahman G. and Shahzad M., Numerical solution of fractional order smoking model via Laplace Adomian decomposition method, Alexandria Engineering Journal, 57 (2018), 1061.

[31] Hassan H. N. and El-Tawil M. A., A new technique of using homotopy analysis method for solving high-order nonlinear differential equations, Math. Method. Appl. Sci., 34 (2011), 728–742.

[32] Himoun N., Abbaoui K. and Cherruault Y., New results of convergence of Adomian's method, Kybernetes 28 (1999), 423–429.

[33] Lin Y., Liu Y. and Li Z., Symbolic computation of analytic approximate solutions for nonlinear differential equations with initial conditions, Computer Physics Communications, 183 (2012), 106.

[34] Mavoungou T. and Cherruault Y., Convergence of Adomian's Method and Applications to Non-linear Partial Differential Equations, Kybernetes, 21 (1992), 13-25.

[35] Ngarhasta N., et al., New numerical study of Adomian method applied to a diffusion model, Kybernetes, 31 (2002) 61–75.

[36] Purohit S. D., Baleanu D. and Jangid K., On the solutions for generalised multiorder fractional partial differential equations arising in physics, Math Meth Appl Sci., 46 (2023), 8139–8147.

[37] Rach R., Adomian G. and R.E. Meyers, A modified decomposition, Comput. Math. Appl., 23 (1992), 17–23.

[38] Shah K. and Bushnaq S., Numerical treatment of fractional endemic disease model via Laplace Adomian decomposition method, Journal of Science and Arts, 39 (2017), 257.

[39] Turkyilmazoglu M., Accelerating the convergence of Adomian decomposition method (ADM), Journal of Computational Science, 31 (2019), 54-59.

[40] Wazwaz A. M., A new method for solving singular initial value problems in the second order ordinary differential equations, Appl. Math. Comput., 128 (2002), 45–57.

[41] Wazwaz A. M., The modified decomposition method for analytic treatment of differential equations, Appl. Math. Comput., 173 (2006), 165–176.

[42] Wazwaz A. M., A new algorithm for calculating Adomian polynomials for nonlinear operators, Appl. Math. Comput., 111 (2000), 53–69.

[43] Wazwaz A. M. and El-Sayed S. M., A new modification of the Adomian decomposition method for linear and nonlinear operators, Appl. Math. Comput., 122 (2001), 393–405.

[44] Zhou Y., Wang M. and Wang Y., Periodic wave solutions to a coupled KdV equations with variable coefficients, Physics Letters A., 308 (2003), 31–36.