APPLYING PARTICLE SWARM OPTIMIZATION BASED ON
PADÉ APPROXIMANT TO SOLVE ORDINARY DIFFERENTIAL
EQUATION

ABDULRAZZAQ T. ABED
AND
AZZAM S. Y. ALADOOL

Department of Mathematics
College of Education for Pure Science
University of Mosul, Mosul, Iraq

(Communicated by Chongyang Liu)

Abstract. Ordinary differential equations are converted into a constrained optimization problems to find their approximate solutions. In this work, an algorithm is proposed by applying particle swarm optimization (PSO) to find an approximate solution of ODEs based on an expansion approximation. Since many cases of linear and nonlinear ODEs have singularity point, Padé approximant which is fractional expansion is employed for more accurate results compare to Fourier and Taylor expansions. The fitness function is obtained by adding the discrete least square weighted function to a penalty function. The proposed algorithm is applied to 13 famous ODEs such as Lane Emden, Emden-Fowler, Riccati, Ivey, Abel, Thomas Fermi, Bernoulli, Bratu, Van der pol, the Troesch problem and other cases. The proposed algorithm offer fast and accurate results compare to the other methods presented in this paper. The results demonstrate the ability of proposed approach to solve linear and nonlinear ODEs with initial or boundary conditions.

1. Introduction. Ordinary differential equations are one of the most important branches of mathematics that are used in many engineering, physical and chemical applications, and it is the basis for their development.

Given their importance, researchers have been interested in finding solutions of ODEs, and seeking for the exact analytic solutions. Although there are many methods that provides an analytic solution. However, there are many types of ODEs that cannot be solved using analytical methods, so they found numerical approach for solving ODEs that are difficult to solve using exact methods, such as Euler’s method, Range Kutta method, Adams-Bashforth and Adams-Moulton methods and others. By looking extensively at numerical methods, each method has a condition to be convergent, and there are several numerical methods that fail to solve nonlinear ODEs [4].
Approximate methods have been established for solving linear and nonlinear ODEs, e.g. Adomian decomposition method (ADM) [3], the differential transformation method (DTM) [19], the variational iteration method (VIM) [2], and the homotopy perturbation method (HPM) [1], etc. These methods have limitations such as assuming conditions for convergence and solving a specific type of nonlinear ODEs.

The researchers presented several methods to solve differential equations based on the intelligent algorithm, the expansion and the fitness function. M. Babaei, introduced PSO based on Fourier expansion and weighted residual function (WRF) [5]. A. Sadollah et al used least squares weighted function (LSWF) as a fitness function via the PSO, Genetic and Cuckoo search algorithms [21]. The Evolutionary algorithms utilized with the even expansion the Fourier series [11]. Taylor expansion method with genetic algorithm is used to solve ODEs [15]. Finally, differential evolution algorithms (SHADE and L-SHADE) with periodic Fourier expansion applied to solve linear and nonlinear ODEs [27].

The Fourier expansion is characterized by being continuous and it can be derived for any order without losing any of its terms. However, it is mainly depends on the number of variables in determining required accuracy. This means increasing the number of variables leads to increase the results accuracy. While this require computational effort and very large storage units. Therefore, it may difficult to reach a satisfactory solution [27].

In general, the results accuracy depends on the search space, the number of variables and the type of extension. So that functions involving singularity points are difficult to approximate using non-singular expansions such as Taylor and Fourier expansion [26].

Therefore Padé expansion is used for the following reasons:

1. The possibility of using \([-1,1]\) as a search space considering the number of variables is applicable.
2. Padé expansion give accurate results with the use of a small number of variable.
3. It is a fractional expansion. So it is used to deal with functions that contain singular-point.

In this paper, the PSO algorithm based on Padé approximant and the discrete least square weighted function (DLSWF) as a fitness function of the minimum optimization problem is presented. Then the stability of the approximate solution and the error arising from it are studied. Furthermore, the results will be compared with the approach of [5] and [15]. In addition a comparison will also be made among WRF, LSWF and DLSWF.

We discuss in this work on 13 famous of linear and nonlinear IVPs and BVPs. Work fundamentals are explained in the next section, followed by the methodology for solving ODEs. The examples is then presented along with the settings, after that the discussion and comparisons are made. Finally the conclusions are drawn.

2. PRELIMINARY. In this section, a simple explanation of linear and nonlinear ODEs is provided. Also the fundamentals of the particle swarm optimization algorithm is showed. Finally the algorithm of PSO is described.

2.1. LINEAR AND NONLINEAR ODEs. Suppose that \(y\) is a function of \(x\) defined on a real interval \([a,b]\). A linear and a nonlinear nonhomogeneous ODEs of order \((n)\) can be written respectively as follows [7], [9]:
\[ y^{(n)} + f_{n-1}(x)y^{(n-1)} + f_{n-2}(x)y^{(n-2)} + \ldots + f_0(x)y = f(x), \quad (1) \]
\[ y^{(n)} + F(x, y, y', \ldots, y^{(n-1)}) = 0, \quad (2) \]
where \( y' = \frac{dy}{dx}, \ n = \frac{d^n y}{dx^n} \) and \( f, f_m, m = 0, 1, \ldots, n-1 \) is a function of \( x \) and \( n \) is a positive integer.

If Eq(1) or Eq(2) subject to the following initial conditions:
\[ y(x_0) = \gamma_0, y'(x_0) = \gamma_1, \ldots, y^{(n-1)}(x_0) = \gamma_{n-1}. \quad (3) \]

Then Eq(1) and Eq(3), Eq(2) and Eq(3) called linear initial value problem (LIVP) and nonlinear initial value problem (NLIVP) respectively.

If Eq(1) or Eq(2) subject to the following boundary conditions:
\[ y(x_0) = \gamma_01, y'(x_0) = \gamma_02, \ldots, y(x_n) = \gamma_n1, y'(x_n) = \gamma_n2. \quad (4) \]

Then Eq(1) and Eq(4), Eq(2) and Eq(4) denoted linear boundary value problem (LBVP) and nonlinear boundary value problem (NLBVP) respectively.

2.2. PARTICLE SWARM OPTIMIZATION. In 1995, J. Kennedy and Russell Eberhart studied the behavior of a group of flocks of birds in an attempt to understand social behavior, they found that the discovered algorithm could be used in optimization problems[13]. The PSO algorithm contains a population of individuals called particles, each particle has a position and velocity that is constantly updating [18]. The movement of each particle is described by the mathematical model[22],[24]:

\[ \xi_i(t + 1) = \omega \times \xi_i(t) + \varsigma_1 \times rand_1 \times (\theta_1(t) - \zeta_i(t)) + \varsigma_2 \times rand_2 \times (\Theta(t) - \zeta_i(t)), \quad (5) \]
\[ \zeta_i(t + 1) = \xi_i(t + 1) + \zeta_i(t), \quad (6) \]
where \( \omega \) is a damping factor, \( \xi_i(t) \) denotes the velocity of the \( i^{th} \) particle of the population at iteration \( t \), \( i = 1, 2, \ldots, \) number of population in swarm (nPop), \( t = 1, 2, \ldots, \) max iteration (Maxit), \( \varsigma_1, \varsigma_2 \) are two acceleration coefficients, \( rand_1, rand_2 \) are two random factors in the interval \([0,1]\), \( \zeta_i(t) \) is position of \( i^{th} \) particle belong in search space \([VarMin, VarMax]\) defined on \( R \), \( \theta_i(t) \) is the best position for \( i^{th} \) particle at time \( t \) and \( \Theta(t) \) is the global best position among all particles.

The PSO algorithm can search in N dimensions according to the number of variables \( nVar \), the values of \( \varsigma_1, \varsigma_2 \) and \( \omega \) are determined as follows [12]:

\[ \phi = \phi_1 + \phi_2, \quad (7) \]
and
\[ \chi = \begin{cases} \frac{2\phi}{\phi^2 - \phi^2 - 4\phi} & \text{if } \phi > 4 \\ \kappa & \text{if } \phi \leq 4 \end{cases}, \quad (8) \]
then
\[ \varsigma_1 = \phi_1 \times \chi, \quad \varsigma_2 = \phi_2 \times \chi, \quad \text{and} \quad \omega = \chi, \quad (9) \]
where \( \kappa \in [0,1] \) and the values of \( \phi_1, \phi_2 \) are randomly selected to investigate the relationship \( \phi > 4 \).

To avoid the non-convergence of PSO, \( \rho \) parameter (defined in \((0,1)\)) is entered to continuously damping the velocity with each iteration and it is applied as follows:

\[ \omega(t + 1) = \rho \times \omega(t). \quad (10) \]
The velocity space is obtained as follows:

\[ \text{MaxVelocity} = \sigma \times (\text{VarMax} - \text{VarMin}), \text{MinVelocity} = -\text{MaxVelocity}, \quad (11) \]

where \( \sigma \) is a positive parameter.

2.3. Algorithms of PSO. In this section, the particle swarm optimization algorithm is detailed, and it is represented by the following steps [23], [14]:

step (1). Picking the value of parameters and inputs: determine \( \phi_1, \phi_2, \kappa, \rho, \sigma, \text{VarMin}, \text{VarMax}, n\text{Pop}, \text{Maxit} \) and \( n\text{Var} \).

step (2). Initialize the velocity: It is entered as a zero vector.

step (3). Initialize the position: It is chosen as a uniform distribution within the search space.

step (4). Evaluate the fitness function of each particle in swarm.

step (5). Obtain personal best and global best: It is calculated by comparing all the results obtained by each particle in the swarm.

step (6). Update position and velocity for each particle: Using Eq (5) and Eq (6).

step (7). Test criteria: If all criteria are succeeded, continue, otherwise go to step(4).

step (8). Stop and give the optimal solution that is the Global best and the Personal best.

3. METHODOLOGY. In this section, the main idea utilized for finding the approximate solution of ODEs are presented. Firstly, we define the expansion function which applied to give an approximate solution. Secondly, the fitness function is set. Furthermore, the penalty function used in achieving the conditions determined by the ODEs is detailed. Finally the algorithm is developed to evaluate an approximate solution of ODEs.

3.1. PADÉ EXPANSION APPROXIMATION (PEA). The approximate solutions of ODEs are generally approximated by an expansion. Likewise, in intelligent algorithms an expansion is used to detect an accurate approximation. The Padé expansion is used as the basis to approximate the solution of ODEs. Suppose that the approximate solution to the ODEs can be written as the expression of Padé expansion, as follows[6]:

\[
y(x) \approx Y_{\text{approx}}(x) = \frac{f(x)}{g(x)} = \frac{\sum_{m=0}^{n_1} \alpha_m x^m}{\sum_{m=0}^{n_2} \beta_m x^m}, \quad (12)
\]

where \( x \) belong to \( I = [x_0, x_n] \) and \( n_1 + n_2 = n\text{Var} \), \( \alpha_m, \beta_m \) are real coefficients which belong to the search space \([\text{VarMin}, \text{VarMax}]\), \( Y_{\text{approx}}(x) \) is the approximate solution, \( y(x) \) is the exact solution and \( g(x) \neq 0 \quad \forall x \in I \).

The first and second derivatives of PEA are calculated as:

\[
y'(x) \approx Y'_{\text{approx}}(x) = \frac{\sum_{m=1}^{n_1} m\alpha_m x^{m-1}}{\sum_{m=0}^{n_2} \beta_m x^m} - \frac{(\sum_{m=0}^{n_1} \alpha_m x^m)(\sum_{m=1}^{n_2} m\beta_m x^{m-1})}{(\sum_{m=0}^{n_2} \beta_m x^m)^2}, \quad (13)
\]
\( y''(x) \approx Y_{\text{approx}}''(x) \)

\[
= \frac{\sum_{m=2}^{n_1} m(m-1)\alpha_m x^{m-2}}{\sum_{m=0}^{n_2} \beta_m x^m} - \left( \frac{\sum_{m=0}^{n_1} \alpha_m x^m \cdot (\sum_{m=2}^{n_2} m(m-1)\beta_m x^{m-2})}{\left( \sum_{m=0}^{n_2} \beta_m x^m \right)^2} \right)
- 2y'(x) \frac{\sum_{m=1}^{n_2} m\alpha_m x^{m-1}}{\sum_{m=0}^{n_2} \beta_m x^m},
\]

(14)

The value of Padé expansion coefficients i.e. \( \alpha_m, \beta_m \) are calculated by applying PSO.

3.2. THE FITNESS FUNCTION. Suppose the exact solution \( y(x) \) of a non-homogeneous ODE with the following boundary or initial conditions which defined on a real interval \( I = [a, b] \) as:

\[
Ly(x) = f(x), \quad Gy(x) = g(x),
\]

(15)

(16)

where \( L \) and \( G \) be differential operator.

Assume the approximate solution is \( Y(x) \), and substitute it in Eq(15) and Eq(16) yields:

\[
LY(x) - f(x) = RL(x), \quad \text{(17)}
\]

\[
GY(x) - g(x) = RG(x), \quad \text{(18)}
\]

where \( RL(x) \) and \( RG(x) \) are residual errors resulting from approximation.

The main objective of this paper is to convert the ODEs into an optimization problem. Therefore, Eq(17) and Eq(18) are formulated as an optimization problem and the goal is to minimize both \( RL(x) \) and \( RG(x) \).

In order to reduce both \( RL(x) \) and \( RG(x) \), a quantitative criterion must be used that determine the accuracy of the approximate solution. The least squares weighted function and the weighted residual function are work as numerical evaluation measure to achieve approximation solution of Eq(17) only. To avoid the problems of calculus and the large computational effort we suggest to compute \( LSWF \) in a discrete form.

Since the PSO algorithm is an unconstrained method, it should be constrained optimization problem. therefore, the penalty function is used for the purpose of satisfying the conditions in Eq(18).

These concepts are briefly presented in present section [5],[21],[28]:

3.2.1. WEIGHTED RESIDUAL FUNCTION (WRF). The equation (17) is satisfied by reducing the following weighted residual function to a minimum:

\[
WRF = \int_D |W(x)| \times |RL(x)| \, dx,
\]

(19)

where \( W(x) \) is the weight function which defined as:

\[
W(x) = \begin{cases} 
1 & \text{if } x \in I \\
0 & \text{if } x \notin I 
\end{cases}
\]

(20)
The ideal case is when \( WRF \) reaches zero, which is rare, so an acceptable error percentage is specified by \( TOL \).

### 3.2.2. LEAST SQUARES WEIGHTED FUNCTION (LSWF)

LSWF is calculated as follows:

\[
LSWF = \int_D |RL(x)|^2 \, dx. \tag{21}
\]

The best approximation is obtained when \( LSWF \) tend to zero or the following condition is met:

\[
\frac{\partial LSWF}{\partial \alpha_m} = 0, \quad \forall \ m = 1, 2, \ldots, nVar, \tag{22}
\]

where \( \alpha_m \) is the unknown coefficients of PEA, TEA or FEA.

Equation (22) can be illustrated as follows:

\[
2 \int_D RL(x) \times \frac{\partial RL(x)}{\partial \alpha_m} \, dx = 0, \quad \forall \ m = 1, 2, \ldots, nVar. \tag{23}
\]

In this case the weight function is defined as:

\[
W_m(x) = \frac{\partial RL(x)}{\partial \alpha_m}, \quad \forall \ m = 1, 2, \ldots, nVar, \tag{24}
\]

### 3.2.3. DISCRETE LEAST SQUARES WEIGHTED FUNCTION (DLSWF)

The DLSWF is calculated as follows:

Dividing the interval \( I \) into \( N \) points \( x_0 = a, x_1, \ldots, x_n = b \), where \( x_k = x_0 + hk \), \( \forall k = 0, 1, \ldots, n \) and \( h > 0 \), and

\[
DLSWF = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (RL(x_k))^2} \tag{25}
\]

### 3.2.4. PENALTY FUNCTION (PF)

The penalty function is used to satisfy all conditions defined by Eq(18) as follows:

\[
PE = \sum_{m=1}^{Nc} K_m |RG(x)|, \tag{26}
\]

where \( K_m, N_c \) are the penalty factor and the number of conditions respectively.

### 3.3. PROPOSED ALGORITHM

In this part, the proposed algorithm for solving ODEs is described in a set of steps as follows:

- **step (1).** Create a Padé expansion approximation: determine the number of coefficients \( nVar \) in array as \( [\alpha_0, \alpha_1, \ldots, \alpha_{n_1}, \beta_0, \beta_1, \ldots, \beta_{n_2}] \).
- **step (2).** Convert the ODE into the implicit form: see Eqs (17) and (18).
- **step (3).** Determine the fitness function (FF) as follows:

\[
FF = DLSWF + PE. \tag{27}
\]

- **step (4).** Apply the PSO algorithm (2.3) to minimize the fitness function.
APPLYING PARTICLE SWARM OPTIMIZATION TO SOLVE ODE

step (5). Calculate the criterion \( FF \) and \( RMSE \). \( RMSE \) is obtained as follows:

\[
RMSE = \frac{1}{N} \sum_{k=1}^{N} |y(x_k) - Y_{approx}(x_k)|
\]  

(28)

step (6). if \( FF \) or \( RMSE < TOL \) STOP and give the PEA, else GOTO step(4).

4. ODEs EXAMPLES AND SETUP. In this section, the examples of discussion are introduced. Then a setting for each example is designed in terms of parameters, device type and the software used to obtain the algorithm.

4.1. TEST EXAMPLES. In this paper, first and second order of 13 famous ODEs are included in table(1)[8],[20],[16],[10],[25],[17].

4.2. SETUP AND PROGRAM OF APPROACH. The agreed parameter and inputs values for all examples are set in table(2). The search space and the number of variables of PEA, TEA and FEA for each example are given in Table(3). The reason behind choosing these values is to ensure satisfactory results for all expansions. The comparison for all examples are carried out in terms of accuracy, time and number of repetitions. It should be noted that increasing the number of variables require to increase the number of iterations, the number of population in PSO and increase or decrease the space of research.

The integrals is calculated by Trapezium method. A judgement is based on comparing the number of iterations, consumed time, and required accuracy. The algorithm reliability is verified by re-running it of 15 times. The algorithm is implemented using Matlab 2018a software, The computer used is a HP laptop, Intel(R) Core(TM) i7-4500U CPU @ 1.80 GHz. 2.40 GHz with RAM 8GB, Windows 10 and system type 64 bit.

5. RESULTS AND DISCUSSION. In this section, an approximate solution of ODEs using Padé approximant with the PSO algorithm and the DLSWF as a fitness function is presented. In addition, the convergence and stability of the algorithm are provided. Furthermore, the results of PSO with Padé, Taylor and Fourier expansions are compared. Moreover, the comparison among WRF, LSWF, DLSWF are drawn.

5.1. PEA-PSO-DLSWF SOLUTIONS. The founded coefficients of ODEs approximate solutions by using PEA with PSO algorithm and the fitness function are listed in Table (4). The plot of each approximate solution corresponding to exact solution for a several examples are shown in Figure (1). Note that the algorithm gives a continuous approximate solution within the given domain. The results indicate that the proposed method gives accurate approximate solutions to ODEs of various levels. The proposed algorithm is succeeded in solving linear and nonlinear ODEs such as Lane Emden, Emden-Fowler, Riccati, Ivey, Abel, Thomas Fermi, Bernoulli, Bratu, Van der pol, the Troesch problem and other problems shown in table (1).
Figure 1. shows a plot of the resulting approximate solution with the corresponding exact solutions.
Table 1. Illustrates different type of the ordinary differential equations, with its conditions, domain and exact solution.

| ODE NAME         | ODE EQUATION       | I.C. or B.C      | DOMAIN | EXACT SOLUTION                |
|------------------|-------------------|-----------------|--------|-------------------------------|
| Simple ODEs1     | \( y' = \frac{y}{x} + 1 \) | \( y(1) = 0 \) | \( x \in [1, 2] \) | \( y = x\ln(x) \)          |
| LIVP1            |                   |                 |        |                               |
| Simple ODEs2     | \( y'' - y = e^x \) | \( y(0) = 0 \) | \( y'(0) = 1 \) | \( x \in [0, 1] \) | \( y = 0.25(e^x + 2xe^x - e^{-x}) \) |
| LIVP2            |                   |                 |        |                               |
| Simple Harmonic  | \( y'' + 25y = 0 \) | \( y(0) = 1 \) | \( y'(\frac{\pi}{10}) = 0 \) | \( x \in [0, \frac{\pi}{10}] \) | \( y = \cos(5x) \) |
| LBVP1            |                   |                 |        |                               |
| Riccati          | \( y' + 1 + 2y - y^2 \) | \( y(0) = 0 \) | \( y'(0) = 1 \) | \( x \in [0, 1] \) | \( y = 1 + \sqrt{2}\tanh(\sqrt{2}x + \frac{1}{2}\ln(x^2 + 1)) \) |
| NLIVP1           |                   |                 |        |                               |
| Abel             | \( y' = 1 - x^2y + y^3 \) | \( y(0) = 0 \) | \( x \in [0, 1] \) | \( y = 1 + \sqrt{2}\tanh(\sqrt{2}x + \frac{1}{2}\ln(x^2 + 1)) \) |
| NLIVP2           |                   |                 |        |                               |
| Ivey             | \( y'' = \frac{(x')^2}{y} - \frac{2x'}{x} - 2y^2 \) | \( y(0) = 1 \) | \( y'(1) = -2 \) | \( x \in [1, 2] \) | \( y = \frac{1}{x^2} \) |
| NLIVP3           |                   |                 |        |                               |
| Lane-Emden       | \( y'' + 2\frac{y'}{x} = -4(2e^y + e^\frac{y}{x}) \) | \( y(0) = 0 \) | \( y'(0) = 0 \) | \( x \in [0, 1] \) | \( y = -2\ln(1 + x^2) \) |
| NLIVP4           |                   |                 |        |                               |
| Emden-Fowler     | \( y'' + 6\frac{y'}{x} + 14y + 4ylny = 0 \) | \( y(0) = 1 \) | \( y'(0) = 0 \) | \( x \in [0, 1] \) | \( y = e^{-x^2} \) |
| NLIVP5           |                   |                 |        |                               |
| Van Der Pol      | \( y'' + y = 0.05(1 - y^2)y' \) | \( y(0) = 0 \) | \( y'(0) = 0.5 \) | \( x \in [0, 1] \) | Sol. By ode45 solver in MATLAB |
| NLIVP6           |                   |                 |        |                               |
| Bratu-Gelfand    | \( y'' - \pi^2e^y = 0 \) | \( y(0) = 0 \) | \( y(1) = 0 \) | \( x \in [0, 1] \) | \( y = -2\ln(\sqrt{2}\cos(\pi(x - 1))) \) |
| NLBVP1           |                   |                 |        |                               |
| Thomas-Fermi     | \( y'' = \sqrt{\frac{x}{\pi}} \) | \( y(1) = 144 \) | \( y(2) = 18 \) | \( x \in [1, 2] \) | \( y = \frac{144}{\pi^2} \) |
| NLBVP2           |                   |                 |        |                               |
| Bernoulli        | \( y'' + (y')^2 = 2e^{-y} \) | \( y(0) = 0 \) | \( y(1) = 0 \) | \( x \in [0, 1] \) | \( y = \log(0.75 + (x - 0.5)^2) \) |
| NLBVP3           |                   |                 |        |                               |
| Troesch Problem  | \( y'' = \sinh y \) | \( y(0) = 0 \) | \( y(1) = 1 \) | \( x \in [0, 1] \) | Sol. By bvp4c solver in MATLAB |
| NLBVP4           |                   |                 |        |                               |

Table 2. the values of parameters and inputs used in all example.

| parameter | \( \phi_1 \) | \( \phi_2 \) | \( \kappa \) | \( \rho \) | \( \sigma \) | \( h \) | \( TOL \) | \( K \) | \( m \) | \( \forall \) | \( Maxit \) | \( nPop \) |
|-----------|--------------|--------------|-------------|---------|---------|------|--------|-------|------|----------|--------|------|
| value     | 2.05         | 2.05         | 1           | 0.8     | 0.4     | 0.01 | 1e-03  | 1 or 10| 500  | 200      |

5.2. **PEA-PSO-DLSWF ERRORS.** The algorithm may be reach to the exact solution. Nevertheless, in many examples the algorithm leaves an error rate made from the approximation. The acceptable error is determined by the value of TOL. The RMSE is obtained to give the error between the exact solution and approximate solution. The RMSE values of proposed approach is about \( 10^{-4} \) to \( 10^{-5} \).
Table 3. Search space and nVar sitting of each example.

| Equation | PEA | TEA | FEA |
|----------|-----|-----|-----|
|          | VarMin | VarMax | nVar | VarMin | VarMax | nVar | VarMin | VarMax | nVar |
| LIVP1    | -1     | 1    | 10   | -2     | 2     | 10   | -2     | 2     | 9    |
| LIVP2    | -2     | 2    | 10   | -2     | 2     | 10   | -2     | 2     | 11   |
| LBVP1    | -3     | 3    | 10   | -3     | 3     | 10   | -1     | 1     | 11   |
| NLIVP1   | -2     | 2    | 10   | -2     | 2     | 10   | -2     | 2     | 9    |
| NLIVP2   | 0      | 1    | 10   | 0      | 1     | 10   | -1     | 1     | 11   |
| NLIVP3   | 0      | 1    | 10   | -3     | 3     | 10   | -2     | 2     | 9    |
| NLIVP4   | -2     | 2    | 10   | -1     | 1     | 10   | -1     | 1     | 11   |
| NLIVP5   | -1     | 1    | 10   | -2     | 2     | 10   | -0.5   | 0.5   | 9    |
| NLBV1    | -1     | 0    | 10   | -3     | 3     | 10   | -1     | 1     | 9    |
| NLBV2    | 0      | 1    | 10   | -900   | 900   | 10   | -200   | 200   | 9    |
| NLBV3    | -2     | 2    | 10   | -3     | 3     | 10   | -1     | 1     | 9    |
| NLBV4    | -4     | 4    | 10   | -1     | 1     | 10   | -1     | 1     | 9    |

Table 4. shows the values of variables founded by using the PSO-PEA-DLSWF algorithm.

| Eq. | Coef. | m = 1     | m = 2     | m = 3     | m = 4     | m = 5     |
|-----|-------|-----------|-----------|-----------|-----------|-----------|
| LIVP1 | α_m  | 0.499646621 | -0.955858566 | -0.662451518 | 0.5990158869 | 0.569745764 |
|      | β_m  | -0.383029549 | 1.0000000000 | 0.9928258613 | 0.2080763899 | -0.01225876 |
| LIVP2 | α_m  | 1.5905e-08  | -2.000000000 | -1.999939132 | -1.9999999202 | 1.320893955 |
|      | β_m  | -2.000000000 | -1.0282027457 | -0.6403099044 | 1.8930067888 | -0.627961135 |
| LBVP1 | α_m  | -0.237082822 | 0.2939052397 | 1.7497711122 | -0.814500817 | -0.25105974 |
|      | β_m  | -0.237082822 | 0.2939052397 | 1.7497711122 | -0.814500817 | -0.25105974 |
| NLIVP1 | α_m  | -6.595e-07  | -2.000000000 | -1.969510526 | -2.000000000 | -1.999999917 |
|      | β_m  | -1.999999998 | 0.1334264437 | -1.999999998 | 0.2828697917 | -1.132644395 |
| NLIVP2 | α_m  | 0.000000000 | 1.000000000 | 1.000000000 | 0.9159523195 | 1.000000000 |
|      | β_m  | 0.000000000 | 1.000000000 | 1.000000000 | 0.9159523195 | 1.000000000 |
| NLIVP3 | α_m  | 1.000000000 | 0.5017899937 | 0.3965523812 | 0.000000000 | 0.000000000 |
|      | β_m  | 0.000000000 | 0.000000000 | 0.000000000 | 0.000000000 | 0.000000000 |
| NLIVP4 | α_m  | -3.556e-07  | 1.1605e-08  | 1.3715434177 | 1.705016485 | 1.664659345 |
|      | β_m  | -0.690913323 | -0.795001437 | -1.237894199 | -0.871324824 | -0.20116878 |
| NLIVP5 | α_m  | -1.000000000 | -0.9999999714 | 1.000000000 | 0.6094151202 | -0.363127581 | 0.471044283 |
|      | β_m  | -0.9999999714 | -1.000000000 | 0.000000000 | 0.000000000 | 0.000000000 |
| NLIVP6 | α_m  | 9.8817e-08  | 0.499999967 | -0.128385779 | -0.225730262 | 0.457988302 |
|      | β_m  | 1.000000000 | -0.333983343 | 0.0548172929 | 0.1919886974 | 0.960184664 |
| LBVP1  | α_m  | 0.0988e-07  | -1.000000000 | -0.168001861 | 0.1680024128 | 1.000000000 |
|      | β_m  | 0.3142717396 | 0.5558835208 | 0.2638355084 | 0.1068186654 | -0.242687286 |
| LBVP2  | α_m  | 1.000000000 | 0.000000000 | 0.000000000 | 0.000000000 | 0.000000000 |
|      | β_m  | 0.000000000 | 0.000000000 | 0.000000000 | 0.000000000 | 0.000000000 |
| LBVP3  | α_m  | -2.279e-11  | -1.512030365 | 0.5287945264 | -0.376567612 | 1.359822951 |
|      | β_m  | 1.512152004 | 0.2390941917 | 1.385962167 | 0.2230063398 | 0.453153062 |
| LBVP4  | α_m  | 2.6966e-06  | -3.398796238 | -0.916386853 | -3.997792887 | -2.079546692 |
|      | β_m  | -1.000000000 | -1.4269898971 | -2.843010822 | -3.999816103 | 1.8712207261 |

in many tests. Figure (2) shows the difference between the exact solution and the approximate solution on defined interval in four examples.

5.3. PEA-PSO-DLSWF CONVERGENCE AND STABILITY. In spite of the number of generations used in the present study is small compared to the previous studies, the approach leads to rapidly convergence. Figure (3) shows the
convergence of the algorithm in all examples using 500 iterations. Note that all the examples with less than 60 iterations reached to acceptable error rate and stable to 500 iterations.

5.4. **COMPARISON BETWEEN PEA-TEA-FEA.** In this part, a numerical comparison of the approximate solutions resulting from PSO - DLSWF based on Pade expansion with solutions based on Taylor and Fourier expansion are presented and summarized in table (5). The comparison is made in terms of time, number of iteration, and RMSE. The bar chart in figure (4) shows the consumed time to obtain solution, each example is represented by three bars PEA, TEA and FEA. Example NLBVP3 is selected randomly to simulate the comparison among PEA, TEA and FEA in terms of convergence, stability and errors(see figure(5)).

The results of using TEA in NLIVP2 consume less time than the use of PEA. However, when TEA is used in (NLIVP3, NLIVP4, NLIVP5, NLBVP1, NLBVP2, NLBVP3, NLBVP4) failed to reach the required accuracy in determined criteria in this work. As a result of this comparison, the proposed method is markedly superior to TEA and FEA.

5.5. **COMPARISON BETWEEN WRF-LSWF-DLSWF.** Fitness function consist of adding one of the following WRF, LSWF, and DLSWF to the penalty function. When boundary or initial conditions are satisfied, the penalty function
becomes zero. Consequently, the remaining part works to improve ODE solution. Table(6) illustrate the numerical comparison between WRF, LSWF and DLSWF by using PEA-PSO algorithm. This shows that DLSWF reduces the computational effort in most examples since WRF and LSWF included integral and calculus forms. The consumed time to find approximate solution, in all example, is shown in bar chart (see figure (6)). Each example is exhibited by three bars WRF, LSWF and DLSWF. Example NLBVP1 is selected to simulate the comparison among WRF, LSWF and DLSWF in terms of convergence, stability and errors(see figure(7)).
Although the accuracy of use DLSWF in this example is less than the use of LSWF. However, DLSWF gives satisfactory results with acceptable error in less time and fewer iteration compared to LSWF. The comparison shows the ability of DLSWF to promote the proposed algorithm with acceptable accuracy and less time consumption in most cases.

6. CONCLUSION. Particle swarm algorithm play a vital role for approximately solving ordinary differential equation. This has been achieved by converting ODEs into constrained optimization problem with the use of approximate expansions. PSO algorithm was applied with the fitness function consisting of DLSWF and the penalty function to find an approximate solution of linear and nonlinear ODEs with initial or boundary conditions. The proposed algorithm was based on the fractional expansion which was Padé approximation to describe approximate solutions. The proposed algorithm applied to solve 13 famous problems. Results of the proposed

| Equation | PEA | TEA | FEA |
|----------|-----|-----|-----|
| LIVP1    | 0.047 | 0.098e-04 | 20.79 |
| LIVP2    | 0.017 | 0.035e-04 | 10.37 |
| LBVP1    | 0.019 | 0.049e-04 | 09.25 |
| NLIVP1   | 0.049 | 0.053e-04 | 25.92 |
| NLIVP2   | 0.006 | 0.053e-04 | 03.61 |
| NLIVP3   | 0.006 | 0.083e-04 | 09.67 |
| NLIVP4   | 0.028 | 0.080e-04 | 28.07 |
| NLIVP5   | 0.008 | 0.076e-04 | 07.12 |
| NLIVP6   | 0.014 | 0.079e-04 | 18.14 |
| NLBV1    | 0.014 | 0.090e-04 | 08.89 |
| NLBV2    | 0.013 | 0.089e-04 | 12.08 |
| NLBV3    | 0.014 | 0.084e-04 | 16.40 |
| NLBV4    | 0.013 | 0.058e-04 | 10.62 |

Figure 4. Comparison of expansions: PEA, TEA and FEA to consumed time.
Figure 5. shows the simulating of example (NLBVP3) in terms of errors, convergence and stability.

Table 6. Comparison of fitness function: WRF, LSWF and DLSWF. W, L and D symbols indicates that method PEA-PSO-WRF, PEA-PSO-LSWF and PEA-PSO-DLSWF is the best results respectively.
Figure 6. Comparison between the fitness function: WRF, LSWF and DLSWF with consumed time.

(a) Convergence and Stability  
(b) Errors  
(c) Plotting

Figure 7. shows the simulating of example (NLBVP1) in terms of errors, convergence and stability.
approach were compared with methods based on Taylor and Fourier expansion. DLSWF is proposed to compute LSWF in a discrete form to avoid the integration computation effort. A comparison was made between WRF, LSWF, and DLSWF. The outcomes of this work highlighted the effectiveness of the use of the proposed algorithm for solving ODEs as showed promoting results in terms of convergence, stability, accuracy, consumed time, and number of iterations.

Acknowledgements. The authors write to express thanks to the technical and academic staff of the University of Mosul as well as the College of Education for Pure Science for their support to facilitate this work.

REFERENCES

[1] S. Abbasbandy, Homotopy perturbation method for quadratic Riccati differential equation and comparison with Adomian’s decomposition method, *Applied Mathematics and Computation*, Elsevier, 172 (2006), 485–490.

[2] E. J. Ali, New treatment of the solution of initial boundary value problems by using variational iteration method, *Basrah Journal of Science, Basrah University*, 30 (2012), 57–74.

[3] M. Almazmumy, F. A. Hendi, H. O. Bakodah and H. Alzumi, Recent modifications of Adomian decomposition method for initial value problem in ordinary differential equations, *American Journal of Computational Mathematics*, Scientific Research Publishing, 2 (2012), 228–234.

[4] U. M. Ascher, R. M. M. Mattheij and R. D. Russel, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Society for Industrial and Applied Mathematics, 1995.

[5] M. Babaei, A general approach to approximate solutions of nonlinear differential equations using particle swarm optimization, *Applied Soft Computing*, Elsevier, 13 (2013), 3354–3365.

[6] G. A. Baker and P. Graves-Morris, *Padé Approximants*, Second edition: Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 1996.

[7] A. Borzí, *Modelling with Ordinary Differential Equations: A Comprehensive Approach*, CRC Press, 2020.

[8] W. E. Boyce, R. C. DiPrima and D. B. Meade, *Elementary Differential Equations and Boundary Value Problems*, 11th edition, WILEY, 2017.

[9] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th edition, Brooks/Cole, Cencag Learning, 2011.

[10] S. Chakraverty and Susmita Mall, *Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations*, CRC Press, 2017.

[11] J. M. Chaquet and E. Carmona, Solving differential equations with Fourier series and evolutionary strategies, *Appl. Soft Comput.*, 12 (2012), 3051–3062.

[12] M. Clerc and J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation*, IEEE, 6 (2002), 58–73.

[13] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, IEEE, (1995), 39–43.

[14] A. Engelbrecht, Particle swarm optimization: Velocity initialization, 2012 IEEE Congress on Evolutionary Computation, IEEE, (2012), 1–8.

[15] D. Gutierrez-Navarro and S. Lopez-Aguayo, Solving ordinary differential equations using genetic algorithms and the Taylor series matrix method, *Journal of Physics Communications*, IOP Publishing, 2 (2018), 115101.

[16] M. Hermann and M. Saravi, *Nonlinear Ordinary Differential Equations: Analytical Approximation and Numerical Methods*, Springer India, 2016.

[17] E. A. Hussain and Y. M. Abdul–Abbass, Solving differential equation by modified genetic algorithms, *Journal of University of Babylon for Pure and Applied Sciences*, 26 (2018), 233–241.

[18] J. Kennedy, The particle swarm: social adaptation of knowledge, Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC’97), IEEE, (1997), 303–308.

[19] F. Mirzaee, Differential transform method for solving linear and nonlinear systems of ordinary differential equations, *Applied Mathematical Sciences*, 5 (2011), 3465–3472.
[20] T. Radhika, T. Iyengar and T. Rani, Approximate Analytical Methods for Solving Ordinary Differential Equations, CRC Press, 2014.

[21] A. Sadollah, H. Eskandar, D. G. Yoo and J. H. Kim, Approximate solving of nonlinear ordinary differential equations using least square weight function and metaheuristic algorithms, Engineering Applications of Artificial Intelligence, Elsevier, 40 (2015), 117–132.

[22] A. Sahu, S. K. Panigrahi and S. Pattnaik, Fast convergence particle swarm optimization for functions optimization, Procedia Technology, Elsevier, 4 (2012), 319–324.

[23] M. Shehab, A. T. Khader and M. Al-Betar, New selection schemes for particle swarm optimization, IEEE Transactions on Electronics, Information and Systems, The Institute of Electrical Engineers of Japan, 136 (2016), 1706–1711.

[24] S. Talukder, Mathematical Modelling and Applications of Particle Swarm Optimization, Ms. S. Thesis, Blekinge Institute of Technology, School of Engineering, 2011.

[25] W. F. Trench, Elementary Differential Equations with Boundary Value Problems, Brooks/Cole Thomson Learning, 2013.

[26] W. Van Assche, Padé and Hermite-Padé approximation and orthogonality, arXiv:math/0609094, 2006.

[27] Z. Zhang, Y. Cai and D. Zhang, Solving ordinary differential equations with adaptive differential evolution, IEEE Access, IEEE, 8 (2020), 128908–128922.

[28] P. Zitnan, Discrete weighted least-squares method for the Poisson and biharmonic problems on domains with smooth boundary, Appl. Soft Comput., 217 (2011), 8973–8982.

Received June 2020; 1st revision December 2020; Final revision January 2021.

E-mail address: abdulrazzaq.1990@uomosul.edu.iq
E-mail address: Azzam.aladool@uomosul.edu.iq