International Journal of Scientific Research in Civil Engineering
© 2019 IJSRCE | Volume 3 | Issue 1 | ISSN : 2456-6667

Optimization of Civil Engineering's Initial Value Problems by Particle Swarm Optimization Algorithm

Fatima OUAAR*, Naceur KHELIL
Department of Mathematics, University Mohamed KHEIDER, Biskra, Algeria

ABSTRACT

A differential equation is a mathematical equation that relates some function with its derivatives. In applications, the functions usually represent physical quantities, the derivatives represent their rates of change, and the equation defines a relationship between the two. Because such relations are extremely common, differential equations play a prominent role in many disciplines including engineering, physics, economics, and biology. They are useful in many domains. Traditionally, Initial Value Problems (IVPs) in Ordinary Differential Equations (ODEs) can be solved by classical mathematical methods, which are not very precise namely in the difficult problems. Swarm Intelligence Algorithms (SI) are considerate as a crucial factor of modern optimization when large sort of Nature Inspired Algorithms have emerged recently to treaty successfully a variety of problems. In this paper, Particle Swarm Optimization algorithm (PSO) is used to solve approximately an (IVP), by a selected example; the efficiency of the considered method is verified by means of a simulation study compared by a Runge Kutta method that shows very good results.

Keywords: Optimization Problems, Initial-Value Problem (IVP); Runge Kutta method; Particle Swarm Optimization Algorithm (PSO).

I. INTRODUCTION

Differential equations have wide applications in various engineering and science disciplines. In general, modelling of the variation of a physical quantity, such as temperature, pressure, displacement, velocity, stress, strain, current, voltage, or concentration of a pollutant, with the change of time or location, or both would result in differential equations. Similarly, studying the variation of some physical quantities on other physical quantities would also lead to differential equations.

In fact, many engineering subjects, such as mechanical vibration or structural dynamics, heat transfer, or theory of electric circuits, are founded on the theory of differential equations. It is practically important for engineers to be able to model physical problems using mathematical equations, and then solve these equations so that the behaviour of the systems concerned can be studied.

Let $f=f(x,y)$ be a real-valued function of two real variables defined for $a \le x \le b$, where a and b are finite, and for all real values of y. The equations

$$\begin{cases} y' = f(x,y) \\ y(a) = y_0 \end{cases} \quad (1)$$

are called first order Initial Value Problem (IVP); they represent the following problem: To find a function $y(x)$, continuous and differentiable for $x \in [a,b]$ such that $y'=f(x,y)$ from $y(a)=y_0$ for all $x \in [a,b]$ [12]. This problem have unique solution if: f is continuous on

$[a,b]\times\mathbb{R}$, and satisfies the Lipschitz condition; it exists a real constant $k>0$, as $|f(x,\theta_1)-f(x,\theta_2)|\leq k|\theta_1-\theta_2|$, for all $x\in[a,b]$ and all couple $(\theta_1,\theta_2)\in\mathbb{R}\times\mathbb{R}$.

Finding the optimal solutions numerically of an IVP is gotten with approximations: $y(x_0+h),\ldots,y(x_0+nh)$ where $a=x_0$ and $h=(b-a)/n+1$. For more precision of the solution, we must use a very small step size h that includes a larger number of steps, thus more computing time which is not available in the useful numerical methods like Euler and Runge-Kutta methods [12], that may approximate solutions of IVP and perhaps yield useful information, often sufficing in the absence of exact, analytic solutions.

In pure mathematics, differential equations are studied from several different perspectives, mostly concerned with their solutions—the set of functions that satisfy the equation. Only the simplest differential equations are solvable by explicit formulas; however, some properties of solutions of a given differential equation may be determined without finding their exact form.

If a self-contained formula for the solution is not available, the solution may be numerically approximated using computers. The theory of dynamical systems puts emphasis on qualitative analysis of systems described by differential equations, while many numerical methods have been developed to determine solutions with a given degree of accuracy.

Biological, physical, or chemical systems in nature are the subject of many nature-inspired meta-heuristic algorithms [20] [21] [23]. Swarm Intelligence has become trendy among researchers working on optimization problems all over the world [11] [13] it demonstrates their capabilities in solving many optimization problems by taking a dissimilar forms according to the inspired process of the natural systems like Genetic algorithm [10], Ant colony optimization algorithm [5], Bee algorithm [16], Bat algorithm [22], etc. All these algorithms have several advantages illustrated via a wide range of applications.

The Particle Swarm Optimization algorithm (PSO) [14][15] [3] is a bio-inspired optimization, which takes an interesting place between nature inspired algorithms [8][9] maintained by its nice performance against several classical meta-heuristic algorithms [7]. This is behind the vast utilizations of PSO in various domains such as chemical engineering, civil engineering, electrical and communication engineering, computer science, etc [1] [4] [6] [25] [24]. PSO was modified and hybridized with other nature inspired meta-heuristic algorithms in order to overcome its limitations [17] [18] [19].

In this paper, IVP is formulated as an optimization problem, and then the PSO is used as a tool to find numerical solutions for this problem compared by the Runge Kutta method.

This paper is organized as follows: The formulation of the problem is revealed in section 2; section 3 explains the Runge Kutta method, section 4 provides basics on PSO and its main steps for finding an approximate solution of IVP. The Section 5 exposes numerical results when by chosen example to show how the PSO can lead to a satisfactory result for solving IVP. The comments and conclusion are made in section 6.

II. PROBLEM FORMULATION

A. Objective function

The main idea in the formulation of the objective function is to use the finite difference formula for the derivative and equation (1) we obtain,

$$\frac{y(x_j)-y(x_{j-1})}{h}\approx f\left(x_{j-1},y\left(x_{j-1}\right)\right).$$

Thus,

$$\frac{y_j - y_{j-1}}{h} \approx f(x_{j-1}, y_{j-1}).$$

Consequently, we have to consider the error formula:

$$\left[\frac{y_j - y_{j-1}}{h} - f(x_{j-1}, y_{j-1})\right]^2$$

The objective function, associated to Y= $(y_1, y_2, \dots, y_d)$ will be:

$$F(y) = \sum_{i=1}^{d} \left[\frac{y_j - y_{j-1}}{h} - f(x_{j-1}, y_{j-1})\right]^2 \qquad (2)$$

B. Consistency

We are interested in the calculation of *Y=(*y_1*, y2,…,* y_d *)* which minimizes the objective function in equation (2). We have from Taylor's formula order 1:

$$y_j = y_{j-1} + hy'_{j-1} + O(h^2), \qquad j = 1, \dots, d.$$

So,

$$\frac{y_j - y_{j-1}}{h} = y'_{j-1} + O(h)$$

If we subtract $f(x_{j-1}, y_{j-1})$ from both sides of last equation, we obtain:

$$\frac{y_j - y_{j-1}}{h} - f(x_{j-1}, y_{j-1}) = y'_{j-1} - f(x_{j-1}, y_{j-1}) + O(h) \quad ,$$ j=1,…,d.

The last relation shows that the final value Y=(y1,y2,…, y_d) is an approximate solution of IVP, for small value of *h*.

III. RUNGE-KUTTA METHOD

Let an IVP be specified as follows [12]:

$$y' = F(t, y), y(t_0) = \hat{y}(t_0) \qquad (3)$$

where

$$y = (y_1, y_2, \dots, y_n)$$

and

$$F = (y_2, \dots, y_n, f(t, y_1, y_2, \dots, y_n))$$

are vector-valued functions.

Classical Runge-Kutta method is an important iterative method for approximating the solution *y= y(t)* of the IVP (3) by the following equations:

$$\begin{cases} y(t_{n+1}) = y(t_n) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ t_{n+1} = t_n + h \end{cases}$$

where h is the step length, and

$$\begin{cases} k_1 = F(t_n, y(t_n)) \\ k_2 = F(t_n + \frac{1}{2}h, y(t_n) + \frac{1}{2}hk_1) \\ k_3 = F(t_n + \frac{1}{2}h, y(t_n) + \frac{1}{2}hk_2) \\ k_4 = F(t_{n+1}, y(t_n) + hk_3) \end{cases}$$

Thus, the next value $y(t_{n+1})$ is determined by the present value $y(t_n)$ plus the weighted average of four increments k, where each increment is the product of the size of the interval, h, and an estimated slope specified by function F on the right-hand side of the differential equation.

IV. PARTICLE SWARM OPTIMIZATION ALGORITHM

In PSO the set of particles is called a swarm [14] [15] [3]. A swarm consists of N particles moving around in a D-dimensional search space. The position of the i^{th} particle can be represented by:

$$x_i = (x_{i_1}, x_{i_2}, \dots, x_{i_D}).$$

The velocity for the i^{th} particle can be written as:

$$v_i = (v_{i_1}, v_{i_2}, \dots, v_{i_D}).$$

The positions and velocities of the particles are confined within $[X_{min}, X_{max}]$ and $[V_{min}, V_{max}]$, respectively. Each particle coexists and evolves

simultaneously based on knowledge shared with neighboring particles, it makes use of its own memory and knowledge gained by the swarm as a whole to find the best solution. The best previously encountered position of the i^{th} particle is denoted its individual best position

$$x_i^p = (x_{i_1}^p, x_{i_2}^p, \ldots, x_{i_D}^p).$$

a value called p_i^{best} . The best value of the all individual p_i^{best} values is denoted the global best position

$$x_i^g = (x_{i_1}^g, x_{i_2}^g, \ldots, x_{i_D}^g).$$

and called g^{best}. The PSO process is initialized with a population of random particles, and the algorithm then executes a search for optimal solutions by continuously updating generations. At each generation, the position and velocity of the i^{th} particle are updated by p_i^{best} and g^{best} in the swarm. The update equations can be formulated as:

$$v_i^{new} = w \times v_i^{old} + c_1 \times r_1 \times \left(p_i^{best} - x_i^{old}\right) + c_2 \times r_2 \times (g_i^{best} - x_i^{old}), \quad (4)$$

$$x_i^{new} = x_i^{old} + v_i^{new}, \quad (5)$$

where r_1 and r_2 are random numbers between [0, 1], and c_1 and c_2 are acceleration constants, which control how far a particle will move in a single generation. Velocities v_i^{new} and v_i^{old} denote the velocities of the new and old particle, respectively. x_i^{old} is the current particle position, and x_i^{new} is the new, updated particle position. The inertia weight w controls the impact of the previous velocity of a particle on its current one [8][9].

The typical pseudo-code of the PSO process is shown below.

Table 1. Pseudo code of PSO algorithme

PSO pseudo-code

Begin;
Randomly initialize particles swarm;
While (number of iterations, or the stopping criterion is not met);
Evaluate the Fitness of particle swarm;
For n = 1 to number of particles;
Find p^{best};
Find g^{best};
For d = 1 to number of dimension of particle
Update the position of particles by Eq. (4) and (5);
Next d;
End;
Next n;
End;
Next generation until stopping criterion;
End;

V. EXPERIMENTAL RESULTS

Let us take a simple IVP of the type

$$\begin{cases} y'(x) = 1 - y(x) \\ y(0) = 0 \end{cases} \qquad 0 \le x \le 10 \quad (6)$$

This example can be viewed as typical case which provides a good illustration. We note that, the accuracy of results depends manifestly to success of particles in the swarm to locate the best points. For easy interpretation, the numerical results evaluated by PSO algorithm, and those obtained by the Runge Kutta rule have been compared via Figure 1 and Figure 2. The best partition generated by PSO are displayed in Tables 1 and 2 for the cases N=5 and N=11 respectivelly. For convenience, we have presented the parameters settings to generate the PSO algorithm for both cases in Table (3).

Table 2. A partition describing the best partition X generated by PSO algorithm N=4

i	XRK4	YRK4	XPSO	YPSO
0	0.0000	0.0000	0.0000	0.0000
1	2.5000	0.3500	1.7000	0.8173
2	5.0000	0.5795	5.0000	0.9933
3	7.5000	0.7274	8.5000	0.9998
5	10.000	0.8232	10.000	1.0001

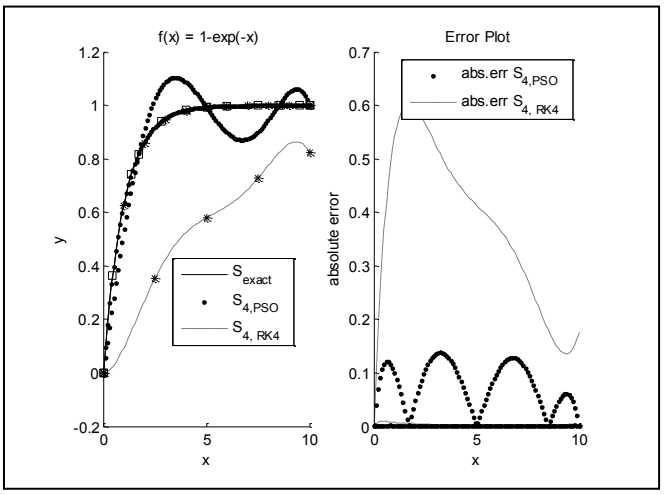

Figure 1. Solution of the equation (6), for N=4. Solid line: exact solution, dots line: RK4, short broken curve (square) : PSO result

Table 3. A partition describing the best partition X generated by PSO algorithm N=10

i	XRK4	YRK4	XPSO	YPSO
0	0.0000	0.0000	0.0000	0.0000
1	1.0000	0.6250	0.4500	0.3624
2	2.0000	0.8594	1.3500	0.7408
3	3.0000	0.9473	2.8000	0.9392
4	4.0000	0.9802	4.0000	0.9817
5	5.0000	0.9926	5.0000	0.9933
6	6.0000	0.9972	6.0000	0.9975
7	7.0000	0.9990	7.5000	0.9994
8	8.0000	0.9996	8.6000	0.9998
9	9.0000	0.9999	9.5000	0.9999
10	10.000	0.9999	10.000	1.0000

Table 4. Parameters setting to generate the PSO algorithm for this example

	N=5	N=11
P opulation size	21	21
Number of iterations	500	600
Accelerations coefficients:c1andc2	0.5	0.5
Inertia weight	1.2 to 0.4	1.2 to 0.4
Desired accuracy	10^{-2}	10^{-3}

The comparison between the performances of PSO and Runge Kutta face to the exact results confirm that PSO is better than Runge Kutta because its curve is closer to the exact results curve contrary to Runge Kutta method.

The absolute error study between the exact results and the studied methods outcomes shows that PSO method offers a very negligible absolute error compared to Runge Kutta

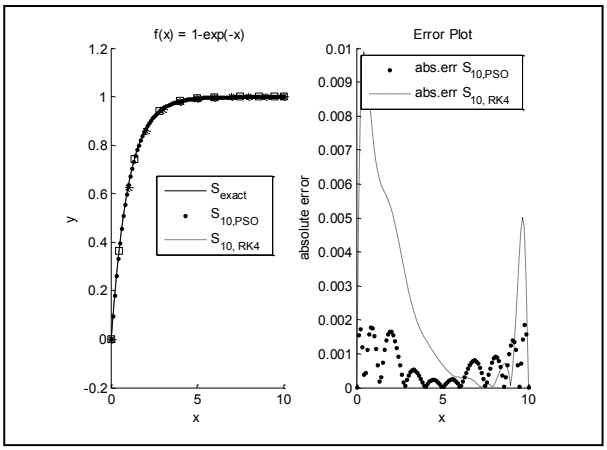

Figure 2. Solution of the equation (5), for N=10. Solid line: exact solution, dots line: RK4, short broken curve (square): PSO result

VI. CONCLUSION

In this study, we apply the PSO to solve approximately an (IVP), via a chosen example and after a comparison between the exact solutions, the algorithm outcomes and Runge Kutta method results;

PSO was found better by offering accurate solutions with smallest amount error.

REFERENCES

[1]. Cagnina L.C, Esquivel S.C., and Coello C.A., "Solving engineering optimization problems with the simple constrained particle swarm optimizer", Informatica 2008, Vol. 32, pp.319-326.

[2]. Clerc, M., "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization", Proceedings of the 1999 IEEE Congress on Evolutionary Computation, Washington DC, 1999, pp.1951—1957.

[3]. Cristian, T. I., "The particle swarm optimization algorithm: convergence analysis and parameter selection", Information Processing Letters, Vol. 85, No. 6, 2003, pp.317-325.

[4]. Djerou, L., Khelil, N., and Batouche, M., "Numerical Integration Method Based on Particle Swarm Optimization" Y. Tan et al. (Eds.) : Part I, LNCS 6728, Springer-Verlag Berlin Heidelberg, 2011, pp. 221-226.

[5]. Dorigo, M., Maniezzo, V., and Colorni, A., "The ant system: optimization by a colony of cooperating agents". IEEE Trans. Syst. Man Cybern1996; B 26, pp.29—41.

[6]. Eberhart, R.C., and Shi, Y., "Parameter selection in particle swarm optimization", in Porto, V.W., 1998.

[7]. Eberhart, R.C. and Kennedy, J., "A new optimizer using particles swarm theory" Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp.39--43,

[8]. Fourie, P.C., and Groenwold, A.A., "Particle swarms in size and shape optimization," Proceedings of the International Workshop on Multi-disciplinary Design Optimization, Pretoria, South Africa, August 7—10, 2000, pp.97—106.

[9]. Fourie, P.C., and Groenwold, A.A., "Particle swarms in topology optimization" Extended Abstracts of the Fourth World Congress of Structural and Multidisciplinary Optimization, Dalian, China, June 4—8, 2001, pp.52, 53.

[10]. Goldberg, D.E., "Genetic Algorithms in Search. Optimization and Machine Learning". Addison Wesley, 1989.

[11]. Holland, J.H., "Adaptation in Natural and Artificial Systems". University of Michigan Press, 1975.

[12]. Henrici, P., "Elements of Numerical Analysis, " New York: McGraw-Hill, 1964.

[13]. Kennedy, J., and Eberhart, R.C., "Swarm Intelligence, " Morgan Kaufmann Publishers, San Francisco, 2001.

[14]. Kennedy, J., and Eberhart, R.C., "Particle swarm optimization". In: Proceedings of IEEE International Conference on Neural Networks 1995; No. IV. 27 Nov--1 Dec, pp. 1942--1948, Perth Australia.

[15]. Kennedy, J., "The behaviour of particles, " Evol. Progr. VII, , 1998. pp.581-587.

[16]. Nakrani, S., Tovey C., "On honey bees and dynamic allocation in an internet server colony". Adapt. Behav, 2004; 12(3--4). 223—240.

[17]. Parsopoulos, K.E., and Vrahatis, M.N., "Modification of the Particle Swarm Optimizer for Locating all the Global Minima", V. Kurkova et al., eds., Artificial Neural Networks and Genetic Algorithms, Springer, New York, 2001, pp.324-327 .

[18]. Shi, Y.H., and Eberhart, R.C., "Fuzzy adaptive particle swarm optimization," IEEE Int. Conf. on Evolutionary Computation, 2001 , pp.101-106.

[19]. Shi, Y.H., and Eberhart, R.C., "A modified particle swarm optimizer," Proc. of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, 1998.

[20]. Yang, X.S., (2010), "Engineering Optimization: An Introduction with Metaheuristic Applications", Wiley, New York.

[21]. Yang X.S., "Book Nature Inspired Optimization Algorithm", Elsevier, 2014.

[22]. Yang, X.S., Gandomi A.H., "Bat algorithm: a novel approach for global engineering optimization", Eng. Comput 2012, Vol.29, Num.5, 464—483.

[23]. Yang, X.S., "Nature-Inspired Metaheuristic Algorithms", Luniver Press, 2008.

[24]. Zerarka, A., and Khelil, N., "A generalized integral quadratic method: improvement of the solution for one dimensional Volterra integral equation using particle swarm optimization," Int. J. Simulation and Process Modelling, 2(1-2), 2006, pp.152-163.

[25]. Zerarka, A., Soukeur, A., Khelil, N., "The particle swarm optimization against the Runge's phenomenon," Application to the generalized integral quadrature method, International Journal of Mathematical and Statistical Sciences, pp-171-1761 :3 2009.