Input Variant Particle Swarm Optimization for Solving Ordinary and Partial Differential Equations with Constraints

Ahmad Haj Mosa, Jean Chamberlain Chedjou, Mouhannad Ali and Kyandoghere Kyamakya
University of Klagenfurt
9020 Klagenfurt, Universttstrasse 65-67, Austria
Email: forename.surname@aau.at

***Abstract*—In this paper, a novel method for solving ordinary and partial differential equations is presented. The proposed method gives a numerical solution based on an input variant particle swarm optimization; where each particle, has a dimension equal to the number of expected solutions. The proposed method also overcomes the problem of considering many constraints (initial and boundary conditions) for the solution.The main motivation of this paper, is to find a solver that is accurate, fast and can handle many constrains as well as many variables. This method is tested over many systems of ordinary and partial differential equations. Further, the proposed is compared with some leading state of the art techniques.**

I. Introduction

Ordinary (ODEs) and partial (PDEs) differential equations are widely used to model many problems in engineering and physics. This justifies the fact, that many approaches are developed to solve systems of ODEs or PDEs. Some methods use analytical solution [1]and the final solution is expressed into algebraic form. A recent sample of analytical solver is solving differential equations using artificial neural network [2], [3]. This technique gives an accurate solution with low usage off memory space. In addition, it can be implemented in neuro or parallel processors. However, it relies highly on the size of the training set and restricts to few solution constraints (initial and boundary conditions). Another techniques of solving differential equations is the numerical solution [4], [5], [6]. Numerical analysis represents the solution in the form of a vector of values, whose element index represents the system input (e.g. time step). (Garcia,2006) presents a numerical solution using an optimization approach. This method, overcomes the problem of solution constraints. However, This classical method is costly, since its optimization function requires high number of decision variables. The main goal of this paper is to propose a technique that overcomes the underlined drawbacks of the classical methods. This motivation is embodied in finding a solver that is accurate, fast and flexible to the constraints problem. As a result of that, this paper proposes a new solver. This solver models the target problem as an input variant optimization approach. Indeed, it optimizes the solution at each input step via Particle Swarm Optimization (PSO), whose number of decision variables is equal to the number of solutions. The proposed method can be used to solve linear, nonlinear, homogeneous and non homogeneous ODEs or PDEs. The method also overcomes the problem related to constraints. This method that we denoted DEPSO is solving Differential Equations using PSO.

II. Description of the method

The general definition of a system of differential equations is used in this section to describe the proposed method DEPSO, whose explicit form is giving by:

$$\vec{F}(t, Y(t), \nabla Y(t), \dots, \nabla^n Y(t)) = \vec{0} \quad t \; \varepsilon \; D \tag{1}$$

where $\vec{F} = (F_1, F_2, \dots, F_M)^T$; $\vec{0} = (0, 0, \dots, 0)^T$ a zeros vector with size M; $Y(t) = (y_1(t), y_2(t), \dots, y_K(t))$; $\nabla = d \; or \; \partial$ denote ordinary or partial derivative; D is the input set. The first step is to transform the general form given by (1) to an optimization problem at input t in order to find the solution $Y(t)$. The general form of the optimization function is given by:

$$\min G(\vec{F}, \vec{BC}, t) \tag{2}$$

where

$$G(\vec{F}, \vec{BC}, t) = \sum_{m=1}^{M} F_m^2(t) + \sum_{r=1}^{R} BC_r^2(t) \tag{3}$$

$\vec{BC} = (BC_1(Y(t)), BC_2(Y(t)), \dots, BC_R(Y(t)))^T = \vec{0}$ is the vector of boundary conditions. Then, the goal is to find the minimum value (should be very close to 0) of $G(\vec{F}, \vec{BC}, t)$ at each input step t, where the decision variables are the solution at each input t. This means at each input step, there are n decision variables represented by the vector $\vec{o} = (y_1^*(t), y_2^*(t), \dots, y_n^*(t))^T$. The optimization process is then applied for all input values t. The second step is to find a fast and reliable optimization method.Therefore, Particle Swarm Optimization is chosen [8]. In order to use PSO, the problem addressed in Eq.2 has to be discretized. Therefore, finite difference method [4] is used to express the cost function G into discrete form; h is the discretization step size.

III. Particle Swarm Optimization (PSO) as a slover

Recently PSO becomes one of the most important optimization problems solver [8]. The basic idea behind PSO is to make the optimizer rely on the collaboration between many particles, and thereby creating the so called swarm. These particles search through different directions around the searching space in order to find the optimum solution. PSO is composed of number of iterations. At each iteration i the particle p has a position vector $\vec{Y}_p^i$ and a velocity vector $\vec{V}_p^i$. The velocity vector updates the particle position to a new direction. This direction is influenced by the best position the related particle has reached and the best position among all particles. This update process is performed using the following equations [9], [10]:

$$\begin{aligned}\vec{V}_p^i = \omega \vec{V}_p^{i-1} + c_1 rd(L_p^{i-1} - \vec{Y}_p^{i-1}) \\ +c_2 rd(L_{global}^{i-1} - \vec{Y}_p^{i-1})\end{aligned} \tag{4}$$

$$\vec{Y}_p^i = \vec{Y}_p^{i-1} + \vec{V}_p^i \tag{5}$$

where $rd\ \varepsilon[0\ 1]$ is a random number; c_1 and c_2 are the self confidence operators; ω is the inertia weight [9] which varies between 0.9 and 0.4 with respect to direction of search; L_p^{i-1} is the best position for particle p; L_{global}^{i-1} is the best position among all particles. In DEPSO, the position and velocity of each particle are given by

$$\vec{Y}_p^i[t_s] = (y_{p,1}^i[t_s], y_{p,2}^i[t_s], \ldots, y_{p,k}^i[t_s])^T \tag{6}$$

$$\vec{V}_p^i[t_s] = (v_{p,1}^i[t_s], v_{p,2}^i[t_s], \ldots, v_{p,k}^i[t_s])^T \tag{7}$$

where t_s is the discrete time obtained with step size h.

IV. Results

In this section, an example of DEPSO is presented. This example represents a system of two first order ODEs described by the following equations:

$$\frac{dy_1}{dt} = \cos(t) + y_1^2(t) + y_2(t) - (1 + t^2 + \sin^2(t)) \tag{8}$$

$$\frac{dy_2}{dt} = 2t - (1 + t^2)\sin(t) + y_1(t)y_2(t) \tag{9}$$

The initial conditions are $y_1(0) = 0$ and $y_2(0) = 1$. Using finite difference method with an interval h, The discrete forms of Eq(9) and Eq(10) are given by

$$\begin{aligned}F_1[t_s] = \frac{y_1[t_s+1] - y_1[t_s]}{h} - \cos(t_s h) + y_1^2[t_s] + y_2[t_s] \\ -(1 + (t_s h)^2 + \sin^2(t_s h))\end{aligned} \tag{10}$$

$$\begin{aligned}F_2[t_s] = \frac{y_2[t_s+1] - y_2[t_s]}{h} - 2t_s h - (1 + (t_s h)^2) \\ \sin(t_s h) + y_1(t_s)y_2(t_s).\end{aligned} \tag{11}$$

Thus, the optimization cost function is given by

$$G(\vec{F}, t_s) = F_1^2[t_s] + F_2^2[t_s]. \tag{12}$$

The next process is to optimize G at each step t_s using PSO to obtain the solution vector $\vec{o} = (y_1^*, y_2^*)$. Fig 1 shows the result of DEPSO (red line) with comparison to the result using Matlab solver ode113 (blue line). Where $h = 0.01$ and $ts = [0\ 500]$ a discrete form of $t = [0\ 5]$.

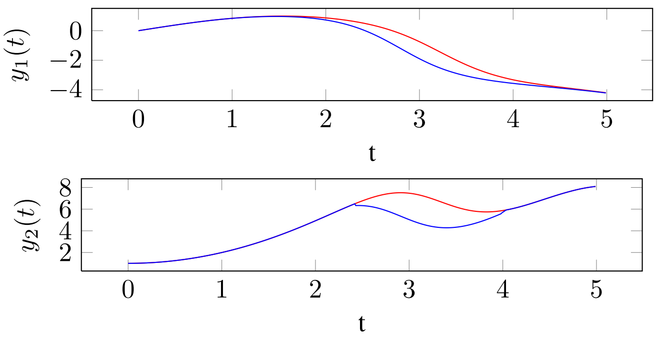

Fig. 1.
Evaluation of DEPSO: Matlab solver (Red), DEPSO (Blue)

V. Conclusion

A novel method is presented for solving ordinary and partial differential equations. This method overcomes the problem of having many boundary conditions as well as many variables. The proposed method can be used to solve linear, non linear, homogeneous and non homogeneous differential equations. The main idea behind this method, is to transform the differential equation problem to an input variant optimization problem. Therefore, we propose an input variant particle swarm optimization method to get the solution of the target differential equation.

References

[1] Murray, Speigel *Schaum's Outline of Laplace Transforms*, McGraw-Hill; 1 edition (June 1, 1965)

[2] Lagaris, I.E.; Likas, A.; Fotiadis, D.I.; *Artificial neural networks for solving ordinary and partial differential equations*, Neural Networks, IEEE Transactions on , vol.9, no.5, pp.987-1000, Sep 1998.

[3] Khan, J.A.; Zahoor, R.M.A.; Qureshi, I.M. *Swarm Intelligence for the Solution of Problems in Differential Equations*, Environmental and Computer Science, 2009. ICECS '09. Second International Conference on , vol., no., pp.141-147, 28-30 Dec. 2009

[4] Delfour, M.; Fortin, M.; and Payre, G; *Finite difference solutions of a non-linear schrdinger equation*, Journal of Computational Physics, 44:277-288, 1981

[5] Fei, Z.; Garcia, P.V; and Vazquez, L; *Numerical simulation of nonlinear schrdinger systems*, A new conservative scheme. Applied Mathematics and Computation, 71:165-177, 1995

[6] Johnson, C.; *Numerical Solution of Partial Differential Equations by the Finite Element Method (Dover Books on Mathematics*, Dover Publications (January 15, 2009)

[7] Garcia, R.; Garcia, V.M.P; *Solving Functional and Differential Equations with Constraints via an Optimization Approach*, International Conference on Mathematical and Statistical Modelling, Castillo, 2006

[8] Kennedy, J and Eberhart, R *Particle Swarm Optimization*, proc proceedings of the IEEE International Conference on Neural Network, Perth, Australia, IEEE Service Center, Vol. 4. Piscataway, NJ, 1995, pp.1942-1948

[9] Shi, Y.; Eberhart, R.; *A modified particle swarm optimizer*, Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on , vol., no., pp.69-73, 4-9 May 1998

[10] Ratnaweera, A.; Halgamuge, S.K.; Watson, H.C.; *Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients*, Evolutionary Computation, IEEE Transactions on , vol.8, no.3, pp. 240- 255, June 2004