Open Journal of Optimization, 2015, 4, 69-73
Published Online September 2015 in SciRes. http://www.scirp.org/journal/ojop
http://dx.doi.org/10.4236/ojop.2015.43009

Solving Ordinary Differential Equations with Evolutionary Algorithms

Bakre Omolara Fatimah[1], Wusu Ashiribo Senapon[2], Akanbi Moses Adebowale[2]

[1]Department of Mathematics, Federal College of Education (Technical), Lagos, Nigeria
[2]Department of Mathematics, Lagos State University, Lagos, Nigeria
Email: larabakre@yahoo.com, wussy_ash@yahoo.com, akanbima@gmail.com

Received 2 June 2015; accepted 1 September 2015; published 4 September 2015

Abstract

In this paper, the authors show that the general linear second order ordinary Differential Equation can be formulated as an optimization problem and that evolutionary algorithms for solving optimization problems can also be adapted for solving the formulated problem. The authors propose a polynomial based scheme for achieving the above objectives. The coefficients of the proposed scheme are approximated by an evolutionary algorithm known as Differential Evolution (DE). Numerical examples with good results show the accuracy of the proposed method compared with some existing methods.

Keywords

Evolutionary Algorithm, Differential Equations, Differential Evolution, Optimization

1. Introduction

For centuries, Differential Equations (DEs) have been an important concept in many branches of science. They arise spontaneously in physics, engineering, chemistry, biology, economics and a lot of fields in between. Many Ordinary Differential Equations (ODEs) have been solved analytically to obtain solutions in a closed form. However, the range of Differential Equations that can be solved by straightforward analytical methods is relatively restricted. In many cases, where a Differential Equation and known boundary conditions are given, an approximate solution is often obtainable by the application of numerical methods.

Several numerical methods (see [1]-[3]) have been developed to handle many classes of problems but yet, the quest for reasonably stable, fast and more accurate algorithms is still on the search in the field of calculus.

Since many evolutionary optimization techniques are methods that optimizing a problem by iteratively trying

How to cite this paper: Fatimah, B.O., Senapon, W.A. and Adebowale, A.M. (2015) Solving Ordinary Differential Equations with Evolutionary Algorithms. *Open Journal of Optimization*, **4**, 69-73. http://dx.doi.org/10.4236/ojop.2015.43009

to improve a candidate solution with regard to a given measure of quality (see [4]-[7]), interest in the adaptation of these techniques to Differential Equations is recently on the rise. Approximate solutions of Differential Equations are obtained by formulating the equations as optimization problems and then solved by using optimization techniques.

Nikos [8] in his work proposed the idea of solution of ODEs via genetic algorithm combined with collocation method. In [6], the combination of genetic algorithm with the Nelder-Mead method was introduced and implemented for the solution of ODEs and the idea of neural network for obtaining approximate solutions of ODEs was also proposed in [9]. The author in [10] adapted the classical genetic algorithm to the solution of Ordinary Differential Equation.

In this paper we show that the Differential Evolution (DE) algorithm can also be used to find very accurate approximate solutions of second order Initial Value Problems (IVPs) of the form

$$y'' + p(t)y' + q(t)y = r(t); \quad y(t_0) = y_0, \ y'(t_0) = y_0', \quad t \in [t_0, b] \tag{1}$$

2. Basic Notions of Differential Evolution Algorithm

Formally, let $f : \mathbb{R}^n \to \mathbb{R}$ be the function which must be optimized. The function takes a candidate solution as argument in the form of a vector of real numbers and produces a real number as output which indicates the fitness of the given candidate solution. The gradient of f is not known. The goal is to find a solution m for which $f(m) \le f(p)$ for all p in the search-space, which would mean m is the global minimum. Maximization can be performed by considering the function $g := -f : \mathbb{R}^n \to \mathbb{R}$ instead.

Let $\mathbf{x} \in \mathbb{R}^n$ designate a candidate solution (agent) in the population. The basic Differential Evolution algorithm can then be described as follows:

• Initialize all agents $\mathbf{x}$ with random positions in the search-space;

• Until a termination criterion is met (e.g. number of iterations performed, or adequate fitness reached), repeat the following.

• For each agent $\mathbf{x}$ in the population do:

* Pick three agents $\mathbf{a}, \mathbf{b}$, and $\mathbf{c}$ from the population at random, they must be distinct from each other as well as from agent $\mathbf{x}$;

* Pick a random index $R \in \{1, \cdots, n\}$ (n being the dimensionality of the problem to be optimized);

* Compute the agent's potentially new position $\mathbf{y} = [y_1, \cdots, y_n]$ as follows:

• For each i, pick a uniformly distributed number $r_i \equiv U(0,1)$;

• If $r_i < \mathrm{CR}$ or $i = R$ then set $y_i = a_i + F(b_i - c_i)$ otherwise set $y_i = x_i$;

• (In essence, the new position is outcome of binary crossover of agent $\mathbf{x}$ with intermediate agent $\mathbf{z} = \mathbf{a} + F(\mathbf{b} - \mathbf{c})$):

* If $f(\mathbf{y}) < f(\mathbf{x})$ then replace the agent in the population with the improved candidate solution, that is, replace $\mathbf{x}$ with $\mathbf{y}$ in the population.

• Pick the agent from the population that has the highest fitness or lowest cost and return it as the best found candidate solution.

Note that $F \in [0, 2]$ is called the differential weight and $\mathrm{CR} \in [0,1]$ is called the crossover probability, both these parameters are selectable by the practitioner along with the population size $\mathrm{NP} \ge 4$.

3. Construction of Proposed Algorithm

In this section, we show the steps involved in formulating the general linear second order initial value problem (1) as an optimization problem and then use the Differential Evolution algorithm to obtain approximate solution of the ODE.

Consider the second order initial value problem (1), in this work we assume a polynomial solution of the form

$$y(t) = \sum_{i=0}^{k} \psi_i t^i, \quad k \in \mathbb{Z}^+ \tag{2}$$

where ψ_i are coefficients of the monomials t^i to be determined. Substituting (2) and its derivatives into (1) gives

$$\sum_{i=2}^{k} i(i-1)\psi_i t^{i-2} + p(t)\sum_{i=1}^{k} i\psi_i t^{i-1} + q(t)\sum_{i=0}^{k} \psi_i t^{i} = r(t) \tag{3}$$

Using the initial conditions we have the constraint that

$$\left[\sum_{i=0}^{k} \psi_i t^{i}\right]_{t=t_0} = y_0, \quad \text{and} \quad \left[\sum_{i=1}^{k} i\psi_i t^{i-1}\right]_{t=t_0} = y_0' \tag{4}$$

Using (3), at each node point t_n, we require that

$$\mathsf{E}_n(t) = \left[\sum_{i=2}^{k} i(i-1)\psi_i t^{i-2} + p(t)\sum_{i=1}^{k} i\psi_i t^{i-1} + q(t)\sum_{i=0}^{k} \psi_i t^{i} - r(t)\right]_{t=t_n} = 0 \tag{5}$$

To solve the above problem, we need to find the set of coefficients $\{\psi_i | i = 0(1)k\}$, which minimizes the expression

$$\sum_{n=1}^{N} \mathsf{E}_n^2(t) \tag{6}$$

where $N = \dfrac{b-t_0}{h}$ and h is the steplength. We now formulate the problem as an optimization problem in the following way:

$$\text{Minimize: } \sum_{n=1}^{N} \mathsf{E}_n^2(t) \tag{7}$$

$$\text{Subject to: } \left[\sum_{i=0}^{k} \psi_i t^{i}\right]_{t=t_0} = y_0, \quad \text{and} \quad \left[\sum_{i=1}^{k} i\psi_i t^{i-1}\right]_{t=t_0} = y_0' \tag{8}$$

Equations (8) and (9) together is the formulated optimization problem of the IVP (1). The next objective of this work is to solve Equations (8) and (9) using the Differential Evolution algorithm.

Using the Differential Evolution algorithm we are able to obtain the set $\{\psi_i | i = 0(1)k\}$ which minimizes the expression $\sum_{n=1}^{N} \mathsf{E}_n^2(t)$ for each problem. We shall refer to this proposed method as "*Differential Evolution for ODEs* (*DEODEs*)".

4. Numerical Experiments

We now perform some numerical experiments confirming the theoretical expectations regarding the method we have proposed. The propose scheme is compared with the Runge-Kutta scheme for solving (1).

The table of "*CPU-time*" and the maximum error of all computations are also given.

The following parameters are used for all computations.

Differential Evolution:

Cross Probability = 0.5;
Initial Points = *Automatic*;
Penalty Function = *Automatic*;
Post Process = *Automatic*;
Random Seed = 0;
Scaling Factor = 0.6;
Search Points = *Automatic*;
Tolerance = 0.001.

All computations were carried out on a "*Core i3 Intel*" processor machine.

4.1. Problem 1

We examine the following linear equation

$$y''(t) = y'(t); \quad y(0) = 1, \quad y'(0) = 1 \tag{9}$$

with the exact solution $y(t) = \exp(t)$.

Implementing the proposed scheme with $k = 10$, we obtain $\{\psi_i | i = 0(1)10\}$ as

$$\left\{1, 1, \frac{5429956875}{10859913749}, \frac{283916051}{1703496320}, \frac{42051617}{1009238446}, \frac{15735741}{1888307702}, \frac{360751}{259690976}, \frac{80248}{405518915}, \frac{17906}{703880563}, \frac{1622}{708257343}, \frac{202}{441661305}\right\}$$

4.2. Problem 2

Consider the equation

$$y''(t) - 100 = 0; \quad y(0) = 1, \quad y'(0) = -10 \tag{10}$$

with the exact solution $y(t) = 1 - 10t + 50t^2$

Implementing the proposed scheme with $k = 10$, we obtain $\{\psi_i | i = 0(1)10\}$ as

$$\{1, -10, 50, 0, 0, 0, 0, 0, 0, 0, 0\}$$

From the results obtained in **Table 1**, the proposed algorithm gave very accurate coefficients for the solution form for **Problem 1**. The algorithm gave the exact solution for **Problem 2** as seen in **Table 2**.

Table 1. Maximum absolute error and *CPU-time* in seconds for **Problem 1** with step-size $h = 2^{-i}, i = 3(1)9$.

	Maximum Absolute Error		CPU-Time (Seconds)	
i	Runge-Kutta Method	*DEODEs*	Runge-Kutta Method	*DEODEs*
3	4.984042E−6	5.573320E−14	5.210430E−3	4.056000E−4
4	3.281185E−7	6.594725E−14	1.014006E−2	6.864000E−4
5	2.104785E−8	7.016610E−14	2.009293E−2	1.248010E−3
6	1.332722E−9	7.105427E−14	3.996746E−2	2.464820E−3
7	8.383871E−11	7.149836E−14	8.018451E−2	4.836030E−3
8	5.258460E−12	7.149836E−14	1.608682E−1	1.023367E−2
9	3.286260E−13	7.149836E−14	3.238269E−1	2.162174E−2

Table 2. Maximum absolute error and *CPU-time* in seconds for **Problem 2** with steplength $h = 2^{-i}, i = 3(1)9$.

	Maximum Absolute Error		CPU-Time (Seconds)	
i	Runge-Kutta Method	*DEODEs*	Runge-Kutta Method	*DEODEs*
3	0	0	3.182420E−3	2.184000E−4
4	0	0	6.146440E−3	4.056000E−4
5	0	0	1.207448E−2	7.488000E−4
6	0	0	2.421136E−2	1.435210E−3
7	0	0	4.842271E−2	2.870420E−3
8	0	0	9.640862E−2	6.115240E−3
9	0	0	1.964989E−1	1.332249E−2

We see that the Differential Evolution algorithm for solving ODEs gave better approximate results for different steplengths (h) compared with the Runge-Kutta Nystrom method. The proposed solution process also gave better *CPU-Time* for both problems solved.

5. Conclusion

In this paper, we have been able to formulate the general linear second order ODE as an optimization problem, and we have also been able to solve the formulated optimization problem using the Differential Evolution algorithm. Numerical examples also show that the method gives better approximate solutions. Other evolutionary techniques can be exploited as well.

References

[1] Buctcher, J.C. (2008) Numerical Methods for Ordinary Differential Equations. Wiley, New York. http://dx.doi.org/10.1002/9780470753767

[2] Lambert, J.D. (1973) Computational Methods in ODEs. Wiley, New York.

[3] Lambert, J.D. (1991) Numerical Methods for Ordinary Differential Systems. Wiley, New York.

[4] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning. 2nd Edition, Addison-Wesley, Boston.

[5] Holland, H.J. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

[6] Mastorakis, N.E. (2006) Unstable Ordinary Differential Equations: Solution via Genetic Algorithms and the Method of Nelder-Mead. *Proceedings of the* 6*th WSEAS International Conference on Systems Theory & Scientific Computation*, Elounda, 21-23 August 2006, 1-6.

[7] Michalewiz, Z. (1994) Genetic Algorithm + Data Structure = Evolution Programs. 2nd Edition, Springer-Verlag, Berlin. http://dx.doi.org/10.1007/978-3-662-07418-3

[8] Mastorakis, N.E. (2005) Numerical Solution of Non-Linear Ordinary Differential Equations via Collocation Method (Finite Elements) and Genetic Algorithms. *Proceedings of the* 6*th WSEAS International Conference on Evolutionary Computing*, Lisbon, 16-18 June 2005, 36-42.

[9] Junaid, A., Raja, A.Z. and Qureshi, I.M. (2009) Evolutionary Computing Approach for the Solution of Initial Value Problems in Ordinary Diffential Equations. *International Scholarly and Scientific Research & Innovation*, **3**, 516-519.

[10] George, D.M. (2006) On the Application of Genetic Algorithms to Differential Equations. *Romanian Journal of Economic Forecasting*, **2**, 5-9.