Available online at http://scik.org
J. Math. Comput. Sci. 7 (2017), No. 6, 966-972
https://doi.org/10.28919/jmcs/3404
ISSN: 1927-5307

APPROXIMATE SOLUTIONS OF GENERAL SECOND–ORDER INITIAL VALUE PROBLEMS USING DIFFERENTIAL EVOLUTION

OMOLARA FATIMAH BAKRE[1,*], ASHIRIBO SENAPON WUSU[2], MOSES ADEBOWALE AKANBI[2]

[1]Department of Mathematics, Federal College of Education (Technical), Lagos, Nigeria

[2]Department of Mathematics, Lagos State University, Lagos, Nigeria

Abstract. In this paper, it is assumed that the solution of the general second order initial value problem $u'' = f(t,u,u'); \quad u(t_0)=u_0,\ u'(t_0)=u'_0, \quad t \in [t_0,t_n]$ can be approximated by a polynomial *u(t)*. To obtain the values of the coefficients of the terms of *u(t)*, the problem is converted to an optimization problem and the simple stochastic function minimizer called differential evolution is used to obtain the optimal values of the coefficients. Numerical examples show the efficiency and accuracy of the proposed technique compared with some existing classical methods.

Keywords: differential equations; differential evolution; optimization; initial value problems.

2010 AMS Subject Classification: 65L05, 65L06, 65L20, 65L60, 90C27.

1. Introduction

In recent years, the quest for substantially improved accuracy of numerical methods for the solution of ordinary differential equations is on the increase. One of the approach that is gaining

*Corresponding author

E-mail address: larabakre@gmail.com

Received June 21, 2017

more popularity is the idea of converting the differential equation problem to an optimization problem. The transformed problem is then solved using the techniques of optimization. Interestingly, evolutionary algorithms for solving optimization problems appear to be ideal for this type of transformation.

The author in [4] used the classical genetic algorithm to obtain approximate solutions of secondorder initial value problems. Approximate solutions of firstorder initial value problem was computed via the combination of collocation method (finite elements) and genetic algorithms by the author in [6]. In an earlier work, the author in [5] combined the genetic algorithm with the Nelder-Mead method for solving the secondorder initial value problem of the form $u'' = f(x,y)$. The idea of adapting neural network for the solution of secondorder initial value problems was also proposed by the authors in [2]. The use of continuous genetic algorithm for the solution of the twopoint secondorder ordinary differential equation was discussed by the authors in[1]. In an early work by the authors in [7], the adaptation of the differential evolution algorithm for the solution of the secondorder initial value problem of the form $u'' + p(t)u' + q(t)y = r(t)$ was proposed. For the second-order two-point boundary value problem $u'' = f(t,u); \quad u(a) = \eta_1;\ u(b) = \eta_2$ with oscillatory/periodic behaviour, the authors in [3] used the differential algorithm to obtain approximate solutions. In this paper, the second-order initial value problem of the form

$$u'' = f(t,u,u'), \quad u(t_0) = u_0, u'(t_0) = u'_0 \tag{1}$$

is considered. It is assumed that the solution can be approximated by a polynomial *u(t)*. The differential evolution algorithm is used to optimize the coefficients of the terms of the solutions *u(t)*.

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 $f(m) \leq f(p)$ for all p in the search-space, which would mean m is the global minimum. Maximization can be performed by considering the function $h := -f$ 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 $R \in \{1, \ldots, n\}$ (n being the dimensionality of the problem to be optimized).
    - $*$ Compute the agent's potentially new position $\mathbf{y} = [y_1, \ldots, y_n]$ as follows:
      - $\cdot$ For each i, pick a uniformly distributed number $r_i \equiv U(0,1)$
      - $\cdot$ If $r_i <$ CR or $i = R$ then set $y_i = a_i + F(b_i - c_i)$ otherwise set $y_i = x_i$
      - $\cdot$ (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 CR $\in [0,1]$ is called the crossover probability, both these parameters are selectable by the practitioner along with the population size NP ≥ 4.

3. Proposed Method

Consider the second order initial value problem (1), assume a solution of the form

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

where ψ_i are parameters to be determined. Substituting (2) and its derivatives into (1) gives

$$\sum_{i=2}^{k} i(i-1)\psi_i t^{i-2} = f\left(t,u,u'\right) \tag{3}$$

Using the initial conditions we have the constraint that

$$\left[\sum_{i=0}^{k} \psi_i t^{i}\right]_{t=t_0} = u_0, \quad and \quad \left[\sum_{i=1}^{k} i\psi_i t^{i-1}\right]_{t=t_0} = u_0' \tag{4}$$

At each node point t_n, we require that

$$\mathscr{E}_n(t) = \left[\sum_{i=2}^{k} i(i-1)\psi_i t^{i-2} - f\left(t,u,u'\right)\right]_{t=t_n} = 0 \tag{5}$$

To solve the above problem, we need to find the set $\{\psi_i | i = 0(1)k\}$, which minimizes the sum of square of the error at each node point given by the expression

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

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

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

$$\text{Subject to}: \quad \left[\sum_{i=0}^{k} \psi_i t^{i}\right]_{t=t_0} = u_0, \quad and \quad \left[\sum_{i=1}^{k} i\psi_i t^{i-1}\right]_{t=t_0} = u_0' \tag{8}$$

4. Numerical Experiments

To investigate the performance and efficiency of the technique, two numerical test cases are inplemnted. The propose algorithm is compared with the Runge-Kutta Nystrom 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 = 0.55;

Initial Points = Automatic;

Penalty Function = Automatic;

PostProcess = Automatic;

RandomSeed = 0;

ScalingFactor = 0.75;

SearchPoints = Automatic;

Tolerance = 0.000001.

All computations were carried out on a *"Core i5 Intel"* processor machine.

4.1. Problem 1

Consider the initial value problem

$$u''(t) = u'(t); \quad u(0)=1, \quad u'(0)=1, \quad t\in[0,1] \tag{9}$$

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

Implementing the proposed scheme with $k=20$, we obtain $\{\psi_i|i=0(1)20\}$ as

$$\Big\{1,1,\frac{6571423687}{13142847286},\frac{3750870629}{22505274403},\frac{7336207996}{176002735153},\frac{732496906}{91712212181},\frac{705861058}{115866640069},$$
$$-\frac{10986667886}{258057057903},\frac{2823470745}{10261587094},-\frac{1827770759}{1408551750},\frac{25397155139}{5523008714},-\frac{14708743442}{1182364621},$$
$$\frac{35546102816}{1370863545},-\frac{55553531801}{1329747565},\frac{43377764775}{836005046},-\frac{38377792753}{780093189},\frac{61792464421}{1768320675},$$
$$-\frac{47744574869}{2653329647},\frac{20459494229}{3226181133},-\frac{5108479858}{3734187003},\frac{12835075157}{94230559872}\Big\}$$

4.2. Problem 2

Consider the equation

$$u'' = u'+1; \quad u(0)=1, \quad u'(0)=1 \tag{10}$$

with the exact solution $u(t)=2\exp(t)-t-1$

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

$$\Big\{1,1,\frac{10909673950}{10909673949},\frac{853653175}{2560959546},\frac{3343143521}{40117707883},\frac{1184829193}{71090457963},\frac{842174384}{303124757325},$$
$$\frac{185292131}{468169656459},\frac{55944720}{1099597192847},\frac{24891421}{5434233307879},\frac{20840312}{22784206947911}\Big\}$$

	Maximum Absolute Error		CPU-Time (Seconds)	
i	Runge-Kutta Method	*DEODEs*	Runge-Kutta Method	*DEODEs*
3	4.984042E-06	7.580603E-13	1.875000E-2	1.562500E-3
4	3.281185E-07	7.580603E-13	2.968750E-2	3.125000E-3
5	2.104785E-08	7.580603E-13	5.781250E-2	6.250000E-3
6	1.332722E-09	7.747136E-13	1.171875E-1	1.093750E-2
7	8.383871E-11	7.747136E-13	2.453125E-1	2.343750E-2
8	5.258460E-12	7.749357E-13	4.968750E-1	5.625000E-2
9	3.286260E-13	7.767120E-13	9.531250E-1	9.531250E-2

TABLE 1. Maximum Absolute Error and *CPU-time* in seconds for Problem 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	9.968085E-06	1.123546E-13	1.718750E-2	1.562500E-3
4	6.562369E-07	1.261213E-13	2.812500E-2	0.000000E00
5	4.209570E-08	1.327827E-13	6.093750E-2	4.687500E-3
6	2.665442E-09	1.350031E-13	1.125000E-1	7.812500E-3
7	1.676757E-10	1.354472E-13	2.218750E-1	1.406250E-2
8	1.051470E-11	1.354472E-13	5.625000E-1	2.812500E-2
9	6.568079E-13	1.354472E-13	9.250000E-1	6.093750E-2

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

5. Conclusion

In this paper, we have been able to derive and implement an evolutionary scheme for the solution of the second order initial value problem (1) using the techniques of optimimization (Differential Evolution). With the two test problems, clearly, the technique gave very accurate results compared with the well-known Runge–Kutta method for the direct integration of (1).

Also, the technique is stable and consistent even as the stepsize reduces. Other evolutionary techniques can be exploited as well.

Conflict of Interests

The authors declare that there is no conflict of interests.

REFERENCES

[1] A. A. Omar, A. Zaer, M. Shaher, S. Nabil, Solving singular two-point boundary value problems using continuous genetic algorithm, Abstr. Appl. Anal. 2012 (2012), Article ID 205391.

[2] A. Junaid, A. Z. Raja, I. M. Qureshi, Evolutionary Computing Approach for the Solution of Initial Value Problems in Ordinary Diffential Equations, World Acad. Sci. Eng. Tech. 55 (2009), 578-5581.

[3] A. S. Wusu, M. A. Akanbi, Solving oscillatory/periodic ordinary differential equations with differential evolution algorithms, Commun. Optim. Theory 2016 (2016), Article ID 7.

[4] D. M. George, On the appliaction of genetic algorithms to differential equations, Romanian J. Econ. Forecast. 7 (2) (2006), 5-9.

[5] N. E. Mastorakis, Numerical Solution of Non-Linear Ordinary Differential Equations via Collocation Method (Finite Elements) and Genetic Algorithms, Proceedings of the 6th WSEAS Int. Conf. on Eolutionary Computing. Lisbon, Portugal. June 16-18, (2005), 3642.

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

[7] O. F. Bakre, A. S. Wusu, M. A. Akanbi, Solving ordinary differential equations with evolutionary algorithms, Open J. Optim. (4) (2015), 69-73.