Solving ordinary differential equations using genetic algorithms and the Taylor series matrix method

Journal of Physics Communications

OPEN ACCESS

RECEIVED
8 August 2018

REVISED
15 October 2018

ACCEPTED FOR PUBLICATION
2 November 2018

PUBLISHED
13 November 2018

PAPER

Solving ordinary differential equations using genetic algorithms and the Taylor series matrix method

Daniel Gutierrez-Navarro and Servando Lopez-Aguayo

Tecnologico de Monterrey, Escuela de Ingeniería y Ciencias, Ave. Eugenio Garza Sada 2501, Monterrey, N.L., México, 64849

E-mail: servando@itesm.mx

Keywords: genetic algorithms, ordinary differential equations, numerical methods

Abstract

A method for solving ordinary differential equations based in evolutionary algorithms is introduced. The concept of Taylor series matrix is defined, allowing to transform a differential equation into an optimization problem, in which the objective function is constituted by the coefficients of a series expansion. An ad-hoc genetic algorithm is used to find such coefficients that satisfy particular conditions. The efficiency of the algorithm is demonstrated by solving several ordinary differential equations.

1. Introduction

Ordinary differential equations rule a wide spectrum of physical phenomena and they constitute the actual basis of our science and technology, being used to describe a plethora of processes, ranging from quantum mechanics [1], heat transfer [2], electrical circuits, optics [3] and all kinds of harmonic oscillations [4], population growth [5] to cosmology dynamics [6] and even to model nonlinear dynamics of interpersonal relationships [7], just to mention a few examples.

Solving differential equations analytically can take a lot of time, and can be intellectually exhaustive. Even more, some differential equations can not be solved in closed and analytical form and, therefore, numerical methods are often chosen to deal with differential equations [8]. However, the behavior of the numerical methods can be compromised when dealing with non-linearity, higher orders or stiffness, for they can fail completely or accumulate a great deal of numerical error. In this manuscript, an optimization method for solving ordinary differential equations is presented, which can handle general ordinary differential equations with very low numerical error.

A differential equation can be seen as an optimization problem under some circumstances; as an example take the Helmholtz equation that models several phenomena in electromagnetism. It has been shown that this equation can be solved by optimizing the corresponding coefficients of the angular spectrum, instead of dealing directly by the differential equation [9]. In particular, if a solution to a given ordinary differential equation within a well defined domain can be expressed as a series of any orthogonal basis or power series [10], such as:

$$y_{sol}(x) = \sum_{n=0}^{\infty} \alpha_n x^n, \tag{1}$$

then, a fair numerical approximation, is to assume that, for a given range for $x \in [a, b]$, a finite number of coefficients must fulfill the conditions with a minimum numerical error. Thus, the problem may be seen as optimizing the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_c$, such that equation (1) satisfies the differential equation in a particular domain in such way that the residual error is minimized. Here is demonstrated that it is possible to solve this optimization problem of multiple variables by using a genetic algorithm. In general, a genetic algorithm is a computational tool that simulates the process of evolution by means of natural selection, postulated by Charles Darwin, in which a population of possible solutions is submitted to a computational environment which will lead them to converge into an absolute minimum [11]. The flowchart for a conventional genetic algorithm is shown in figure 1. On it, a random-generated group of solutions is created, and only the fittest will survive to

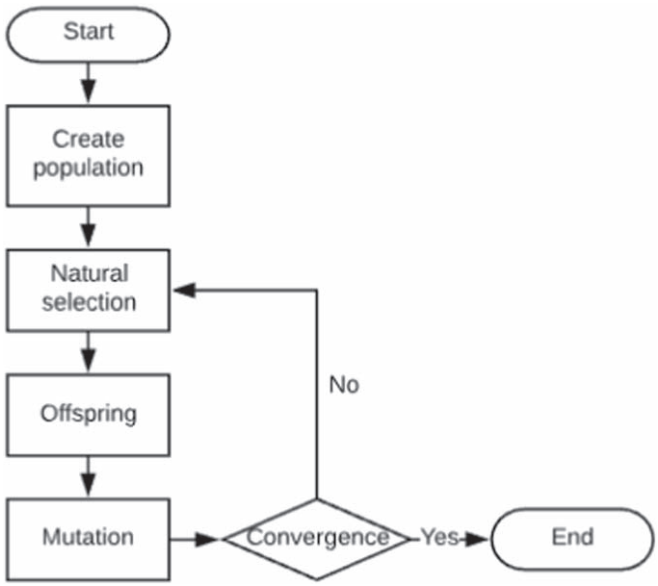

Figure 1. Conventional genetic algorithm flowchart.

reproduce, transmitting information to the next generation of solutions, which will be randomly mutated to avoid falling into local minima.

For the ordinary differential equation case, the coefficients of the orthogonal series expansion will undergo an optimization process given by a genetic algorithm. In this manuscript, an optimization of the Taylor series is reported. However, note that the method is not constrained to this unique kind of series, for it can be used with different orthogonal polynomials or functions [12].

2. The Taylor series matrix

We start by defining the Taylor series matrix as a two dimensional array that includes all the information of the numerical range of the independent variable, as well as the orthogonal nature of the series expansion. Then, as a first step, the numerical resolution of x and $y_{sol}(x)$ is set to have M points, defined by the discretization of the space from a to b, where a is the low domain boundary and b is the high domain boundary. Then x is formulated as a vector array containing M points: $x = [x_1, x_2, \ldots, x_M]$, and the Taylor series matrix is defined as:

$$\mathbb{T} = \begin{bmatrix} x_1^0 & x_1^1 & x_1^2 & \ldots & x_1^{c-1} \\ x_2^0 & x_2^1 & x_2^2 & \ldots & x_2^{c-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_M^0 & x_M^1 & x_M^2 & \ldots & x_M^{c-1} \end{bmatrix}, \tag{2}$$

where c is the total number of coefficients used in the series expansion. Note that this matrix has been conveniently designed in a way that, having a vector array of coefficients $\alpha = [\alpha_0, \alpha_1, \ldots, \alpha_{c-1}]$, then the following point to point product can be easily computed:

$$y_{sol} = \alpha_i * \mathbb{T}_{ij}, \tag{3}$$

where i is the row subindex of all possible coefficients and j corresponds to the column subindex of all solutions. In other words, equation (3) has all the population information of the possible solutions corresponding to the optimization problem, and by a simple point to point matrix multiplication is easy to extract a series expansion solution.

3. The genetic algorithm

As a next step, a particular genetic algorithm is used to find the optimum coefficients that minimize the error in the required ordinary differential equation. The objective of the genetic algorithm is to obtain a function generated by multiplying a vector array of coefficients to the Taylor series matrix, such that a given differential equation is fulfilled.

Let us define

$$Ay'' + By' + Cy = 0; \qquad y(0) = y_0; \qquad y'(0) = y_0', \tag{4}$$

as a general second-order differential equation, where A, B and C are arbitrary coefficients, and y_0 and y_0' are initial condition values.

As a first step for solving the ordinary differential equation by the genetic algorithm, a sample population of pseudo-random-generated coefficients its needed. Hence, every chromosome, or possible solution in the algorithm, is represented as a vector array of c genes (coefficients) $s_n = [\alpha_{0,p}, \alpha_{1,p}, \ldots, \alpha_{c-1,p}]$, where p is the total number of chromosomes, or similarly, the entire population. Thus, s_n is a matrix of $p \times c$ dimension:

$$s_n = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,c} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{p,1} & \alpha_{p,2} & \cdots & \alpha_{p,c} \end{bmatrix}. \tag{5}$$

Given the fact that the range of possible values for the coefficients is *a priori* unknown, an arbitrary range of values can be used, in this case, from a to b. Overall, the sample population matrix can be expressed then as: $s_n = a + (b - a) * \text{rand}(p, c)$. Where rand($p$, c) is a pseudo-random matrix with uniform distribution and values from 0 to 1, which has size $p \times c$. This initial population evolves iteratively with every generation, up to a total number of N generations, following the computational environment dictated by the so-called *fitness* function. A large part of the success for a genetic algorithm comes from the correct design of the corresponding *fitness* function. In this manuscript, the *fitness* function consists in two main parts: the first criteria for solving the problem is that the ordinary differential equation must be satisfied, i.e. the residual error musts tend to zero. Therefore, the function that has to be minimized is (see equation (4)).

$$\text{fitness} = \text{abs}[D^2 * Ay_{sol} + D * By_{sol} + Cy_{sol}], \tag{6}$$

where D is the derivative operator, and y_{sol} is a solution generated by equation (3). However, there is still an infinity number of possible solutions to the general problem and thus, a second criteria must be satisfied that is imposed by the particular initial conditions of the system. In order to fulfill this second criteria, some points in the solution must be fixed. Let the fitness function satisfy

$$\text{fitness} = \gamma * \text{abs}[y_0 - y_{sol}(0)] + (1 - \gamma) * \text{abs}[y_0' - D * y_{sol}(0)], \tag{7}$$

where $\gamma = \sin\left(\frac{n}{N}\right)^2$, n being the number of the present generation and N the number of total generations. Notice that the γ factor helps the algorithm to solve one initial condition at a time, instead of trying to solve both at the same time. This means that the fitness function is not constant throughout the passing generations, meaning that at some point it is solving the first initial condition, at some other point solves the second initial condition, but there are times in which it solves both at the same time. We found that this increases the yield of the algorithm considerably.

The closer equation (7) is to zero, the better the initial conditions are satisfied. Equations (6) and (7) combined, rule the yield of the genetic algorithm. The overall fitness function is:

$$\begin{aligned} \text{fitness} &= \text{abs}[D^2 * Ay_{sol} + D * By_{sol} + Cy_{sol}] \\ &+ \gamma * \text{abs}[y_0 - y_{sol}(0)] + (1 - \gamma) * \text{abs}[y_0' - D * y_{sol}(0)], \end{aligned} \tag{8}$$

Once the population s_n has been evaluated through the fitness function, it is set that only the best half evaluated fitness will survive. It is expected that $p/2$ chromosomes will enter the mating pool s_{mating}. Next, $p/4$ chromosomes are randomly chosen to be the so-called dads, that constitute the first part of the novel chromosomes, and $p/4$ to be the so-called moms, that constitute the second and last part of the novel chromosomes. Having $p/4$ different couples, every couple produce 2 offspring, so that the population number after the reproduction process is p again. After that couples have been formed, offspring must be created and since the numerical range of the coefficients is yet unknown, a traditional standard reproduction scheme, that allows the offspring to exit the artificial boundary that has been given by the initial domain $[a, b]$, is done [13]:

$$s_{\text{offspring}} = \beta(s_{\text{dad}} - s_{\text{mom}}) + s_{\text{dad}}, \tag{9a}$$

$$s_{\text{offspring}} = \beta(s_{\text{mom}} - s_{\text{dad}}) + s_{\text{mom}}, \tag{9b}$$

where β is the reproduction coefficient. This means that the offspring is not limited by the initial domain conditions at all, but they can evolve either within or without that range of values. At the end of the reproduction process, the sample population s_n has grown back to have p chromosomes. From the resultant population, some percentage of genes are randomly mutated. That percentage is dictated by the μ parameter, which we set as a real number between 0 and 1. The mutating genes are randomly selected from the mutation pool, and are replaced in the following way:

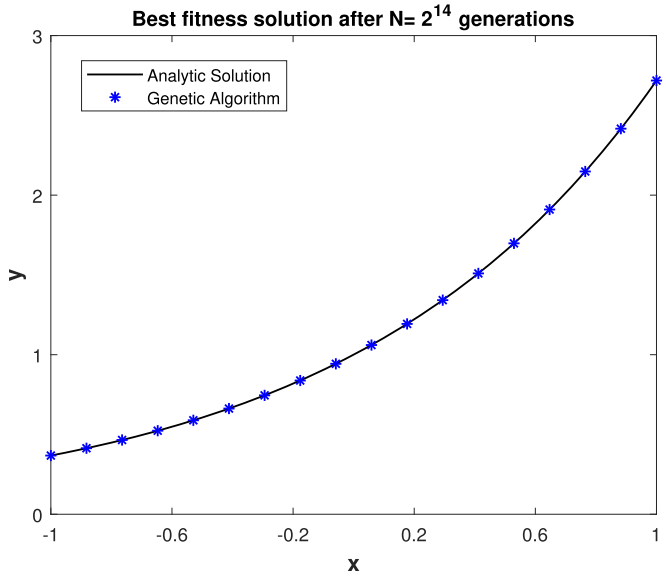

Figure 2. Solution generated by the genetic algorithm to equation (11).

$$s_{\text{mutating}} = a + (b - a) * \text{rand}, \tag{10}$$

where rand is the a pseudo-random number generated, with values from 0 to 1.

After completing the mutation process, the natural selection has ended for that generation. The final population sample s_n will repeat this process up to a total of N generations, and the best evaluated fitness is chosen as the solution for the differential equation.

A quite similar approach has been implemented by Mastorakis [14], and although it has been proven successfully at optimizing the series' coefficients, it has the following limitations: some extra analytical algebra and calculus are needed in order to get the fitness function, it is restricted only to polynomial solutions, and generalization for variable-coefficient differential equations is not an option. Whereas in the Taylor series matrix method the algebra and calculus is done numerically and automatically within the fitness function, another set of orthogonal functions can be used, an example would be Fourier functions, and a generalization to the code for variable-coefficient differential equations can be implemented. Overall the fitness function works different, for in Mastorakis' approach is static, whereas in the Taylor series matrix method is constantly changing.

4. First-order differential equations

As an illustrative example, a simple, yet very common differential equation is chosen to be solved:

$$y' - ky = 0; \quad y(0) = 1, \tag{11}$$

Here k is a real parameter. This equation is well known to model exponential decay or growth in several physical and engineering problems. For simplicity purposes, we set $k = 1$ without loss of generality. The general solution for this differential equation is:

$$y = \exp(x) + c_1, \tag{12}$$

Here c_1 is given by the particular initial conditions and for this case $c_1 = 0$. Note that for first-order differential equations only one initial condition must be satisfied, so the γ factor can be dropped. The proposed fitness function is:

$$\begin{aligned} \text{fitness} = {} & \text{abs}[D * y_{sol} - y_{sol}] \\ & + \text{abs}[1 - y_{sol}(0)], \end{aligned} \tag{13}$$

For given values $N = 2^{14}, p = 258, \beta = 0.6, \mu = 0.1, c = 9, M = 258, a = -1, b = 1$ (all this values remain constant for further demonstrations), the solution is shown in figure 2. It is important to remark, that the values for several of these parameters were chosen after some previous simulations, but it is still an open problem to chose the optimum parameters for developing the fastest genetic algorithm possible.

In order to measure the accuracy of the genetic algorithm, a point-to-point mean squared error scheme is performed in the following way:

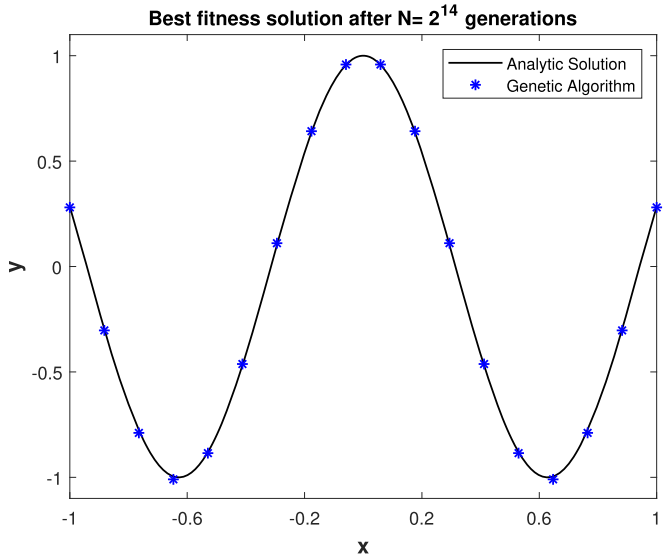

Figure 3. Solution to equation second order differential equation given by equation (15).

$$\mathrm{MSE} = \frac{1}{M}\sum_{m=1}^{M}(y_m - y_{\mathrm{sol}m})^2, \tag{14}$$

where y_m is the analytic solution, $y_{\mathrm{sol}m}$ is the numerical solution given by the algorithm and M is the total number of points used in discretization. For the present example, we find that the numerical error is $\mathrm{MSE} = 3.430\,0 \times 10^{-10}$. Note that all the examples discussed are basically to demonstrate the value and robustness of the numerical method here introduced, and thus, for the general case, where there is not any information about the analytical solution of the corresponding ordinary differential equation, the quantization for the error scheme must be modified. For example, this can be done by changing the step size in the space for several magnitude orders and evaluating the respective change in the residual error, stopping this procedure when the change is less than a certain small control parameter proposed and finally, substituting the best solution obtained into the original differential equation, in order to check the point to point residual error.

5. Second-order differential equations

5.1. Simple harmonic oscillator

As a next example, a solution for one of the most studied physical systems is generated: the simple harmonic oscillator problem, modeled by the following second-order differential equation:

$$y'' + \omega y = 0; \quad y(0) = 1; \quad y'(0) = 0, \tag{15}$$

where we set $\omega = 25$ and thus, the analytic solution is:

$$y = \cos(5x). \tag{16}$$

The fitness function proposed for this problem is then:

$$\begin{aligned}\mathrm{fitness} = {} & \mathrm{abs}[D^2 * y_{sol} + 25y_{sol}] \\ & + \gamma * \mathrm{abs}[1 - y_{sol}(0)] \\ & + (1-\gamma) * \mathrm{abs}[0 - D * y_{sol}(0)],\end{aligned} \tag{17}$$

The solution for this fitness function is shown in figure 3. For this case, the time elapsed was 237.09 s in a standard computer and $\mathrm{MSE} = 7.547\,9 \times 10^{-5}$. Being the genetic algorithm a pseudo-random procedure, it is important to note that it could be necessary to repeat several times the procedure to solve the ordinary differential equations to acquire some representative statistics, especially when we are dealing with not so known ordinary differential equations.

5.2. Damped harmonic oscillator

Next, we show the results for solving the differential equation that corresponds for a particular damped harmonic oscillator problem [15]:

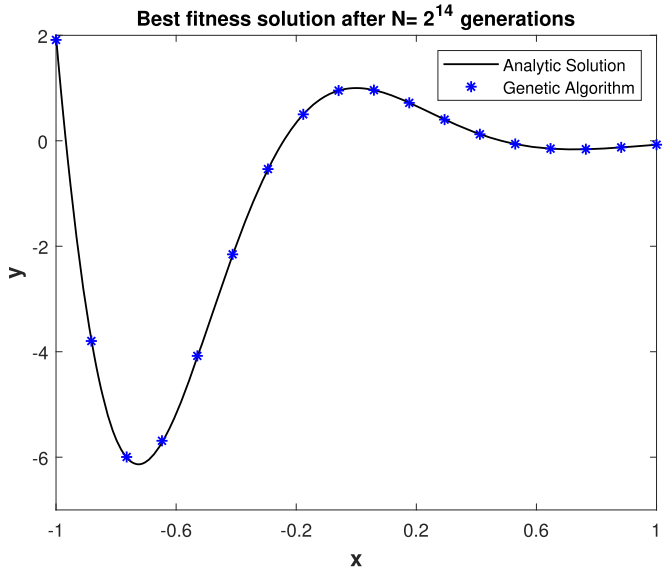

Figure 4. Solution generated by the genetic algorithm to equation (18).

$$y'' + 5y' + 25y = 0; \quad y(0) = 1; \quad y'(0) = 0, \tag{18}$$

which has analytic solution given by:

$$y = e^{-5x/2}\left[\cos\left(\frac{5\sqrt{3}x}{2}\right) + \frac{1}{\sqrt{3}}\sin\left(\frac{5\sqrt{3}x}{2}\right)\right], \tag{19}$$

Where the fitness function for the genetic algorithm to optimize is:

$$\begin{aligned}\text{fitness} = {} & \text{abs}[D^2 * y_{sol} + 5y_{sol} + 25y_{sol}] \\ & + \gamma * \text{abs}[1 - y_{sol}(0)] \\ & + (1 - \gamma) * \text{abs}[0 - D * y_{sol}(0)],\end{aligned} \tag{20}$$

The solution generated by the algorithm is shown in figure 4. The time elapsed was 261.44 s and $\text{MSE} = 2.996\,2 \times 10^{-4}$. Note that in spite of more complexity in the ordinary differential equation than in the simple harmonic oscillator, both systems needed around the same magnitude order in time to be solved.

5.3. Nonlinear ordinary differential equation

Finally, the method here reported is used to demonstrate that is possible to solve a nonlinear ordinary differential equation. For this case, we select a non-linear forced harmonic oscillator problem that is given by:

$$\frac{1}{2}y'' - \frac{1}{2}y + y^3 = 0; \quad y(0) = 1; \quad y'(0) = 0, \tag{21}$$

such equation arrives from reducing the nonlinear-Schrödinger system in order to admit solitons, or nonlinear modes [16]. In fact, this equation has analytic solution given by:

$$y = \text{sech}(x), \tag{22}$$

The fitness function to be optimized for this problem is:

$$\begin{aligned}\text{fitness} = {} & \text{abs}\left[\frac{1}{2}D^2 * y_{sol} - \frac{1}{2}y_{sol} + y_{sol}^3\right] \\ & + \gamma * \text{abs}[1 - y_{sol}(0)] \\ & + (1 - \gamma) * \text{abs}[0 - D * y_{sol}(0)],\end{aligned} \tag{23}$$

The solution generated by the algorithm is shown in figure 5. Note that we are dealing with a problem where the superposition principle does not hold any more, and thus, this is a problem by far more complicated to solve than the past ones.

The time elapsed was 165.60 s and $\text{MSE} = 5.085\,4 \times 10^{-10}$. Note that again, the magnitude order of the time to solve the problem is around the same value of the past examples. This constitutes a strength on the genetic algorithm itself: the procedure here reported can be transparent to certain intrinsic difficulties of very particular ordinary differential equation that is intended to be solved.

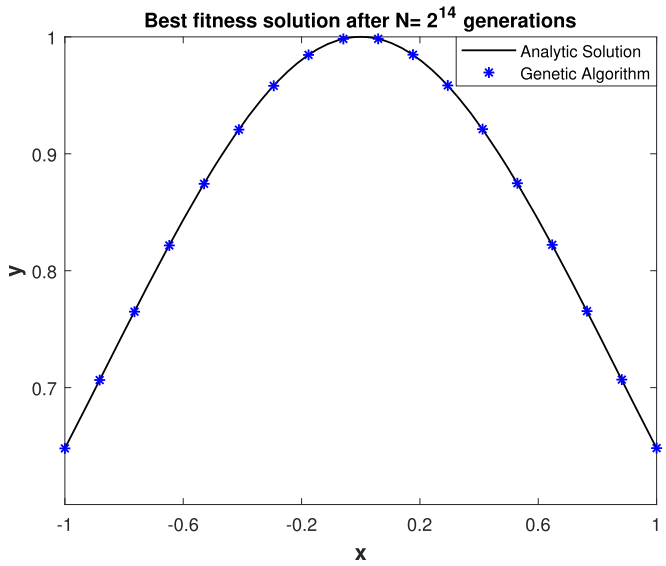

Figure 5. Solution generated by the genetic algorithm to equation (21).

Table 1. Statistical parameters of the algorithm for 100 runs.

Example	Average MSE	MSE variance	Average time elapsed
4.0	3.6886×10^{-10}	3.4219×10^{-20}	145.74 s
5.1	7.551×10^{-5}	1.1359×10^{-13}	309.47 s
5.2	2.9369×10^{-4}	1.9386×10^{-11}	309.33 s
5.3	2.0796×10^{-8}	6.1862×10^{-15}	185.83 s

6. Statistical behavior of the algorithm

In this section we present the long run statistical behavior of the algorithm. A sample of 100 runs is taken from every example presented above, in which the average mean squared error or MSE, MSE variance, and average time is measured. Note that these parameters can be changed by a trade-off between the computational time and the allowed error: thus, it is possible to reduce the time by allowing that less generations evolve in the genetic algorithm, or to reduce the error by using more generations.

It can be appreciated by the results presented in table 1, that the algorithm is consistent in its outcomes.

7. Conclusions

In summary, we report a numerical algorithm: the Taylor-series matrix method, that is based on a genetic algorithm to solve ordinary differential equations by modeling them as optimization problems. The versatility of the genetic algorithm allows the problem to be solved with low numerical error, as it is demonstrated by solving a simple and well known first order equation with exponential solution, the ubiquitous harmonic oscillator equation, the forced harmonic oscillator equation and even a nonlinear ordinary differential equation. We show that it is possible to successfully optimize all the corresponding coefficients in the proposed series solution for all these equations. Thus, the evolutionary algorithm here reported can be used successfully in finding solutions for ordinary differential equations that fulfill conditions of existence and uniqueness. For further development of the algorithm, generalizations must be made in order to be able to change the functions that are used in the expansion of the solution, and the generalization of the algorithm for solving variable-coefficient differential equations and boundary value problems are being worked in with success and the details will be reported in a future manuscript. We hope that this numerical procedure can be used in order to shed light on some physical and engineering problems where ordinary differential equations are highly used.

Acknowledgments

We thank the anonymous reviewers for their very useful and constructive comments. We acknowledge financial support from CONACyT 243284 grant.

References

[1] Namias V 1980 *IMA J. Appl. Math.* **25** 241–65
[2] Pletcher R H, Tannehill J C and Anderson D 2012 *Computational Fluid Mechanics and Heat Transfer* (Boca Raton, FL: CRC Press)
[3] Felix-Rendon U and Lopez-Aguayo S 2017 *J. Opt.* **20** 015606
[4] Edwards C H, Penney D E and Calvis D T 2016 *Differential Equations and Boundary Value Problems* (London: Pearson)
[5] Lefkovitch L 1965 *Biometrics* 1–18
[6] Carloni S, Dunsby P K, Capozziello S and Troisi A 2005 *Class. Quantum Grav.* **22** 4839
[7] Barley K and Cherif A 2011 *Appl. Math. Comput.* **217** 6273–81
[8] Butcher J C 1987 *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods* (New York: Wiley)
[9] Sanchez-Serrano P A, Wong-Campos D, Lopez-Aguayo S and Gutiérrez-Vega J C 2012 *Opt. Lett.* **37** 5040–2
[10] Boas M L 2006 *Mathematical Methods in the Physical Sciences* (New York: Wiley)
[11] Holland J H 1992 *Sci. Am.* **267** 66–73
[12] Arfken G B and Weber H J 1999 *Mathematical Methods for Physicists* (*for Physicists*) (San Diego, CA: Academic)
[13] Haupt R L, Haupt S E and Haupt S E 1998 *Practical Genetic Algorithms* vol 2 (New York: Wiley)
[14] Mastorakis N E 2006 *WSEAS Transactions on Mathematics* **5** 1276
[15] Dekker H 1981 *Phys. Rep.* **80** 1–110
[16] Kivshar Y S and Agrawal G 2003 *Optical Solitons: From Fibers to Photonic Crystals* (New York: Academic)