Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2011, Article ID 521342, 20 pages
doi:10.1155/2011/521342

Research Article

Combining Legendre's Polynomials and Genetic Algorithm in the Solution of Nonlinear Initial-Value Problems

Osvaldo Guimarães, José Roberto C. Piqueira, and Marcio Lobo Netto

Escola Politécnica da Universidade de São Paulo, Avenida Prof. Luciano Gualberto, travessa 3, No. 158, 05508-900 São Paulo, SP, Brazil

Correspondence should be addressed to José Roberto C. Piqueira, piqueira@lac.usp.br

Received 16 January 2011; Revised 9 May 2011; Accepted 24 May 2011

Academic Editor: Wei-Chiang Hong

This work develops a method for solving ordinary differential equations, that is, initial-value problems, with solutions approximated by using Legendre's polynomials. An iterative procedure for the adjustment of the polynomial coefficients is developed, based on the genetic algorithm. This procedure is applied to several examples providing comparisons between its results and the best polynomial fitting when numerical solutions by the traditional Runge-Kutta or Adams methods are available. The resulting algorithm provides reliable solutions even if the numerical solutions are not available, that is, when the mass matrix is singular or the equation produces unstable running processes.

1. Introduction

Mathematical modeling is a constant in engineering daily life, and expressing phenomena by using differential equations is almost mandatory. Successful modeling implies ability to make simplifying assumptions but, even under these assumptions, the solution of differential equations implies a lot of analytical and numerical efforts [1].

Considering these difficulties and sometimes the impossibility of obtaining analytical solutions, several numerical methods were proposed and, nowadays, the great majority of these methods are part of symbolic and numeric softwares as MATLAB and Mathematica, for instance [2]. These methods are accurate and fast, but each one has its limitations, depending on the type of equation to be solved. The main difficulties appear when the mass-matrix is singular or the equation generates instability in the running process [3].

Here, trying to deal with this kind of problem for equations with continuous solutions, a method for initial-value problems in ordinary differential equations is developed, assuming

that the space of solutions is a Hilbert space, equipped with a Legendre's polynomial basis [4].

The central idea is to look for the best polynomial coefficient combination, in order to satisfy the original differential equation in the whole interval, instead of searching a polynomial that better fits a set of points in the given interval.

Consequently, a new error criterion must be defined and a program by using differential evolution methods with elitism [5–7] is developed, based on this new criterion.

The determination of the search space developed here is performed by using sequential contractions, allowing fast evolutions with noticeable reductions in the error bands when compared with random search in large populations, as proposed by [5].

Several examples are solved by the method presented and by using the classical numerical methods, with the accuracy of the results being compared.

In Section 2, the error criterion is discussed and a combination of a global and local criteria is proposed, providing optimal fitting of the polynomial, regarding the function and its derivatives. Section 3 presents the analytical basis of the method, by using Legendre's polynomials to optimize the solutions of an initial-value problem.

The use of the differential genetic algorithm with elitism, guaranteeing stability, applied to the solution of ordinary differential equations, emphasizing the choice of the coefficients under the optimizing criteria developed in Section 2, is described in Section 4. Finally, in Section 5, some examples are presented and the numerical quality of the solutions is discussed.

2. Error Criterion

Usually, the adopted optimization criterion to approximate a function by a polynomial in a given interval is to minimize the square error integral [8]. This kind of approach minimizes the global error along the whole interval, but produces a large value for the local error at the starting and ending points of the interval.

Consequently, when this criterion is applied to an initial-value problem, the approximation does not produce the best result, as the error for the starting point of the interval remains large. To solve this problem, an error criterion considering the local and global results is proposed, minimizing the product of two factors: the maximum error at the starting point of the interval and the global integral error.

To give an idea of this procedure, the differential equation given by (2.1) with initial condition $y(-1) = e^{-1}$, that admits the solution given by (2.2) in the $[-1, 1]$ interval, is studied.

$$\frac{dy}{dx} + 2xe^{-x^2} = 0, \tag{2.1}$$

$$y = e^{-x^2}. \tag{2.2}$$

If one tries to obtain, by using MATLAB, the 8th-order Legendre's polynomial that better fits to (2.2), the result, with double-precision format, is given by (2.3). Considering the substitution of y, given by (2.3), in the original differential equation (2.1), the integral error is $1.9.10^{-8}$ for the whole interval

$$y = .0263x^8 - .1551x^6 + .4963x^4 - .9996x^2 + 1,0000. \tag{2.3}$$

If one continues the process, with the result given by (2.3) as an initial solution candidate to the original equation (2.1), by using the method developed in the next Sections, a solution given by (2.4), with an integral error $8.7.10^{-9}$ for the whole interval, is obtained

$$y = .0242x^8 - .1511x^6 + 0.4940x^4 - .9992x^2 + 1,0000. \tag{2.4}$$

This result is related to the fact that the polynomial given by (2.3) is adequate to fit the function given by (2.2), without considering its derivatives. In order to adequate the solution to the differential equation, given by (2.1), the derivatives must be adjusted in the same way.

3. Analytic Foundations

In this section, it is considered the Hilbert space of piecewise continuous functions, defined in a closed interval $[a,b]$, with a scalar product between f and g, belonging to the space, defined as [4]

$$\langle f, g \rangle = \int_a^b f g \, dx. \tag{3.1}$$

Considering the real interval $[-1,1]$, the Legendre's polynomials set is an orthogonal basis for the space of piecewise continuous function S, defined in this interval, that is, for any function $f \in S$

$$\lim_{n\to\infty} \int_{-1}^{1} \left[f(x) - \sum_{k=0}^{n} a_k P_k(x) \right]^2 dx = 0, \tag{3.2}$$

with P_k being the k-degree normalized Legendre's polynomial and $a_k = \langle f, P_k \rangle$ [4, 9].

It is important to notice that either the integral or the derivative of a Legendre's polynomial can be expressed as a linear combination of the Legendre's polynomial basis [10], as the following theorems, with proofs presented in Appendix A, show.

Theorem 3.1. *Calling $P_n(x)$ the n-degree Legendre's polynomial,*

$$\int_{-1}^{x} P_n(\xi) d\xi = \frac{P_{n+1}(x) - P_n(x)}{2n+1}, \quad n \in N^*. \tag{3.3}$$

Theorem 3.2. *Calling $P_n(x)$ the n-degree Legendre's polynomial,*

$$P_n' = \sum_{k=0}^{n-1} \frac{(2k+1)}{2} \left[1 - (-1)^{k+n} \right] P_k. \tag{3.4}$$

Considering the theorems above, about integration and differentiation of Legendre's polynomials, it is interesting to express these results in a matrix form.

3.1. Operational Matrices

As shown in [11], Theorem 3.3 allows the construction of matrices expressing any linear operation executed over the Hilbert space of the continuous functions. Here, numerical methods for solving differential equations by using Legendre's polynomials are developed. Therefore, constructing these matrices for integration and differentiation operations expressed in a Legendre's polynomials basis is useful.

Let $f(u)$ a linear combination of a generic basis, $\{B_k\}$, in the Hilbert space of continuous functions

$$f(u) = \sum_{k=0}^{n} c_k B_k(u). \tag{3.5}$$

Theorem 3.3. *Let Z a $(n+1)\times(n+1)$ matrix describing the resulting coefficients from a linear operation, α, over the generic basis, $\{B_k\}$, as functions of the same basis. Calling $V = [v_0 v_1 \dots v_n]$ the resulting coefficients vector from the linear operation, α, over $f(u)$: $V^T = Z^T C^T$, with C being the vector of the coefficients described in* (3.5).

The procedure described by Theorem 3.3, proved in Appendix A, can be used for any basis in a Hilbert space, with the calculation of matrix Z representing the main operation to be done.

Combining Theorems 3.1 and 3.3, taking Legendre's polynomials as the basis in the Hilbert space of continuous functions, if Z_{L_i} is $(n+2)\times(n+2)$ and considering α the integral operation

$$\begin{aligned} Z_{L_i}(1,1) &= 1; \qquad Z_{L_i}(1,2) = 1, \\ Z_{L_i}(k,k-1) &= \frac{1}{2k-1}; \quad k = 2,\dots n+1, \\ Z_{L_i}(k,k+1) &= \frac{-1}{2k-1}; \quad k = 2,\dots n+1, \\ Z_{L_i}(n+2,n+1) &= \frac{-1}{2n+3}. \end{aligned} \tag{3.6}$$

All the elements of Z_{L_i}, not defined by (3.6), are equal to zero. For instance, if $n = 2$, Z_{L_i} can be written as

$$Z_{L_i} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ -\frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & -\frac{1}{5} & 0 & \frac{1}{5} \\ 0 & 0 & -\frac{1}{7} & 0 \end{bmatrix}. \tag{3.7}$$

It can be observed that the element $Z_{L_i}(n+2,n+3) = Z_{L_i}(4,5)$ does not appear, because the third degree Legendre's polynomial was not integrated.

Combining Theorems 3.2 and 3.3, taking Legendre's polynomials as the basis in the Hilbert space of continuous functions, if Z_{d_i} is $(n+1)\times(n+1)$ and considering α the derivative operation

$$Z_{L_d}(k+1, j+1) = \left[1-(-1)^{k+j}\right]\frac{2j+1}{2}; \quad k = 1,\dots n-1;\ j = 0,\dots k. \tag{3.8}$$

All the elements of Z_{L_d}, not defined by (3.8), are equal to zero. For instance, for $n = 3$, Z_{L_d} can be written as

$$Z_{L_d} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 1 & 0 & 5 & 0 \end{bmatrix}. \tag{3.9}$$

The integration matrix Z_{L_i} is nonsingular, and, therefore, its inverse always exists. On the other hand, Z_{L_d} needs a correction if c_0 in (3.5) is such that $c_0 \neq -\sum_{n=1}^{k+1} c_n(-1)^n$. The correction to be done is to modify c_0, imposing the condition described, without changing the derivative of $f(u)$.

In [10], the operational matrix was derived for the particular case where α is an integration operation. Theorem 3.3, developed here, permits general matrix expressions for all linear α-operations and considering any kind of basis in the Hilbert space of continuous functions. [12] shows, additionally, this development for Fourier trigonometric and canonical basis.

3.2. Conservation of Initial Conditions in a Legendre's Basis

When using evolution methods to treat initial-value problems, expressing solutions in a Legendre's basis, the initial conditions must be conserved during all algorithm running process. This conservation depends on the correct combination of the coefficients, c_i, to express the function in any step of the running process. This combination generates an algebraic linear system of equations that admits infinite solutions. It is solved choosing a subset of q coefficients, with q representing the order of the original differential equation, and expressing then as a function of the others.

In order to obtain the coefficients of a function derivative, the differentiation matrix, built as explained in the former subsection, has to be used.

As an example, if a third-order equation with solutions in the $[-1, 1]$ interval is studied with seven coefficients for the expression in Legendre's basis, one of the possible relationship among the coefficients is

$$\begin{aligned} c_0 &= 5c_3 - 21c_4 + 56c_5 - 120c_6 + \frac{2}{3}y_0^{(2)} + y_0^{(1)} + y_0, \\ c_1 &= 9c_3 - 35c_4 + 90c_5 - 189c_6 + y_0^{(2)} + y_0^{(1)}, \\ c_2 &= 5c_3 - 15c_4 + 35c_5 - 70c_6 + \frac{1}{3}y_0^{(2)}. \end{aligned} \tag{3.10}$$

where $(y_0, y_0^{(1)}, y_0^{(2)})$ is the vector of the initial conditions. If one needs to obtain the coefficients of the integral of the function, the integration matrix, built as explained in the former subsection, must be used, but the initial condition must be considered, as explained below.

Considering that $F(u) = \int_{-1}^{u} f(\xi)d\xi$ and its Legendres expansion is given by: $F(u) = \sum_{k=0}^{n+1} b_k P_k(u)$, with P_k being the k-degree Legendre's polynomial and that the initial condition for the original differential equation is $F(-1) = \beta$, b_0 obtained by the integration matrix must be replaced by $\overline{b}_0 = b_0 + \beta$. For q differential equations, this process must be repeated q times.

3.3. Domain Transposition

The purpose of this work is to develop numerical methods to solve differential equations with solutions defined in a generic closed real interval [a,b], but the exposed analytical foundations and other developed methods consider only particular intervals [13, 14]. Here, a generic transformation providing tools to transpose any general closed domain to a particular one is studied for up to fourth-order initial-value problems. Besides, the impact of this transformation in original differential equation and initial conditions is analyzed.

Let $u = T(x); T : [a, b] \rightarrow [c, d]$ be a bijective transformation and its inverse T^{-1}, and $y = f(x); f : [a, b] \rightarrow \mathbf{R}$ a solution of an initial value problem, with a vector Y_0 representing the initial condition.

The interval transposed function $y_t(u) : [c, d] \rightarrow \mathbf{R}$ is defined by $y_t(u) = y(x)$, with $x = T^{-1}(u)$ and, for the first derivative, calling $D_0 = y(x)$

$$D_1 = y^{(1)}(x) = y_t^{(1)}(u) \cdot \frac{du}{dx}. \tag{3.11}$$

Calling $\lambda = du/dx = 1/(dx/du)$ and $D_1^t = y_t^{(1)}$,

$$D_1 = D_1^t \lambda. \tag{3.12}$$

Consequently, the second and third derivatives are

$$\begin{aligned} D_2 &= D_2^t \cdot \lambda^2 + D_1^t \cdot \frac{d\lambda}{du} \cdot \lambda, \\ D_3 &= D_3^t \cdot \lambda^3 + 3D_2^t \lambda^2 \frac{d\lambda}{du} + D_1^t \lambda \left[\left(\frac{d\lambda}{du} \right)^2 + \lambda \frac{d^2\lambda}{d^2u} \right]. \end{aligned} \tag{3.13}$$

For the sake of clarity, the forth derivative is written by grouping the terms expressed in function of D_1^t, D_2^t, D_3^t, and D_4^t as follows:

(i) terms in D_4^t: $D_4^t \lambda^4$,

(ii) terms in D_3^t: $6D_3^t \lambda^3 (d\lambda/du)$,

(iii) terms in D_2^t: $D_2^t \lambda^2 (7(d\lambda/du)^2 + 4\lambda(d^2\lambda/du^2))$,

(iv) terms in D_1^t: $D_1^t \lambda [(d\lambda/du)^3 + 4\lambda(d\lambda/du)(d^2\lambda/d^2u) + \lambda^2(d^3\lambda/du^3)]$.

Combining the expressions above,

$$D_4 = D_4^t\lambda^4 + 6D_3^t\lambda^3\frac{d\lambda}{du} + D_2^t\lambda^2\left(7\left(\frac{d\lambda}{du}\right)^2 + 4\lambda\frac{d^2\lambda}{du^2}\right) + D_1^t\lambda\left[\left(\frac{d\lambda}{du}\right)^3 + 4\lambda\frac{d\lambda}{du}\frac{d^2\lambda}{d^2u} + \lambda^2\frac{d^3\lambda}{du^3}\right]. \tag{3.14}$$

As the solution of a differential equation is tied to the initial conditions, it is necessary to establish domain transformations as described above, in an inverse way. Therefore, calling $\rho = dx/du = 1/(du/dx)$ and considering equations (3.12), (3.13), and (3.14) in an inverse way, with the replacement of λ by ρ, the initial conditions, $y(0)$, $y^{(1)}(0)$, $y^{(2)}(0)$, and $y^{(3)}(0)$ are given by

$$\begin{aligned} y_0^t &= y_0, \\ y_1^t &= y_1\rho, \\ y_2^t &= y_2\cdot\rho^2 + y_1\cdot\frac{d\rho}{dx}\cdot\rho, \\ y_3^t &= y_3\cdot\rho^3 + 3y_2\rho^2\frac{d\rho}{dx} + y_1\rho\left[\left(\frac{d\rho}{dx}\right)^2 + \rho\frac{d^2\rho}{d^2x}\right]. \end{aligned} \tag{3.15}$$

If T is a linear transformation, expressions (3.12), (3.13), (3.14), and (3.15), are considerably simplified, as λ and ρ are constants.

In spite of linear transformation being simple, it does not allow to work with equations defined in an infinite domain. In order to take care of this problem, the transformation T, given by

$$\begin{aligned} u = T(x) &= \frac{(d-c)(\arctan(x) - v_1)}{v_2 - v_1} + c, \\ x = T^{-1}(u) &= \tan\left(\frac{(v_2 - v_1)(u-c)}{d-c} + v_1\right), \end{aligned} \tag{3.16}$$

where $v_1 = \arctan a$ and $v_2 = \arctan b$ can be used.

4. Evolution-Adaptive Process for Solving Initial-Value Problems

In this section, the main question to be answered is if it is possible, with the computation power available, to generate random coefficients for an n-degree polynomial and, under a sequential algorithm, like differential evolution algorithm, over the coefficients, to obtain the exact solution of an initial value problem.

Due to the randomness of the original genetic algorithms, the solution space has its regions visited in a random way, and there is no convergence. Nevertheless, when differential versions of genetic algorithms with elitism are applied, some stable processes and satisfactory results are obtained for particular cases [15–18], including complex biological problems [19, 20] and quantum mechanics [21, 22].

Here, a differential genetic algorithm with an implicit learning process, given by a good choice of the region of the space of solutions based on a restriction scheme that continuously contracts the volume of the space of research, guaranteeing stability, is developed.

Considering that a n-degree polynomial is the best candidate to the solution of an initial value problem, in a particular step of a sequential algorithm, under a certain error criterion, it is natural to ask how far of the exact solution it is. In order to treat this problem, the original differential equation is written with the higher-order derivative term expressed as a function of the lower-order derivatives, the original function, and the independent variable as:

$$\frac{d^q \mathrm{y}}{du^q} = G\left(y_t, y_t^{(1)}, \ldots, y_t^{(q-1)}, u\right). \tag{4.1}$$

Replacing the best solution candidate as y, into G given by (4.1), it is possible to evaluate G in the defined interval and fitting the obtained results by using a Legendre's polynomial up to $(n - q)$-degree. The fitting process follows Gauss-Legendre quadrature method [23, 24]. After that, the obtained polynomial is q-times integrated, by using the integration matrix given by $[Z_{L_i}]^T$, defined in Section 3.1, and considering the given initial conditions.

Comparing this result with the original candidate, the dispersion is obtained. This dispersion defines the band variation for the coefficients of the polynomial candidate, in the next step of the algorithm. For the first step, the candidate is the null polynomial, providing the first band variation for each coefficient of the Legendre's polynomial.

To perform the differential evolution algorithm, a population is randomly mounted inside the first band variation. Then, by using a combination of crossing-over, random and mutation, new off-springs are generated feeding the sequence of the algorithm.

The program starts a tournament selection. If the best output of the tournament overcomes the former winner, a new band variation is constructed. Otherwise, by using elitism with random cross-over and mutation [25], the algorithm creates a new generation with each gene (coefficient) varying inside the new band, avoiding stagnation in local minima.

The described procedure is possible, even when the mass-matrix is singular [3] because the Gauss-Legendre's abscissae do not include the interval extremes [23, 24]. If the numerical integration solution is available, it can be the starting point of the algorithm, instead of the null-polynomial, shortening the running time of the procedure.

5. Results and Analysis

Based on the analytical development presented in Section 3, by using a differential evolution algorithm, a method for solving initial value problems, from now on abbreviated by SIV and detailed in Appendix B, was developed, considering that the solutions are expressed as linear combinations of Legendre's polynomials, in the transposed domain.

Starting with an interval transposition, the SIV algorithm runs, adjusting the coefficients of the polynomial solution, minimizing the residual error when replacing the polynomial approximation and derivatives and transposed initial conditions in the transposed differential equation. After a chosen number of generations, applying the inverse transposition, the solution is obtained, in the original domain.

Here, the application of the SIV algorithm is shown considering four examples of nonlinear up to fourth-order equations. The obtained solutions are compared with the analytic ones, when they exist.

The algorithm is implemented with 66 chromosomes and 5% of mutation. The CPU is a PC Dual-core with 2 GB RAM and running MATLAB-7.

In the cases where it is not possible to express an analytic solution by a finite combination of elementary functions, the solution obtained by SIV is compared with the obtained by Adams' method, when the maximum MATLAB precision is used [2, 3].

5.1. Linear Transposition: Zero-Order Bessel Equation

The first chosen example is the zero-order Bessel equation, corresponding to

$$xD_2 + D_1 + D_0 x = 0, \tag{5.1}$$

with D_i denoting the ith derivative of the searched function, related to the independent variable x, $x \in [0,1]$, and with the initial condition given by the vector $[1,0]$.

Equation (5.1) has analytic solution given by

$$J_0(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(k!)^2} \left(\frac{x}{2}\right)^{2k}. \tag{5.2}$$

In order to transpose the domain from the $[0,1]$ to $[-1,1]$, a linear transformation is used. For this kind of transformation, considering that a function is described in an interval $[a,b]$ and will be transposed to an interval $[c,d]$, the relation $(u-c)/(d-c) = (x-a)/(b-a)$, with $u = T(x) = (d-c)/(b-a) \cdot x + (bc-ad)/(b-a)$, is used.

The SIV algorithm was applied to (5.1), with a 26-coefficient polynomial approximation, starting with the null-polynomial.

The final result was compared with the analytic solution (5.2). As, in this case, the mass matrix is singular [3, 23, 24], the Runge-Kutta and Adams algorithms cannot be initialized and, consequently, they are not used for comparisons.

After two hundred generations, an evolved Legendre's series approximation was obtained with very low error margins, as shown in Figure 1 for the obtained solution of (5.1) and the corresponding derivatives.

To give an idea about the precision of the method, Figure 2 shows the residue by substituting the approximation of the function given by (5.2) by using Legendre's polynomial fitting process [2], and the residue for the same degree evolved solution. Observing Figure 2, it can be noticed that the SIV algorithm improves the accuracy in, at least, two order of magnitude. Besides, the problem of lack of accuracy at the interval extremes is considerably reduced.

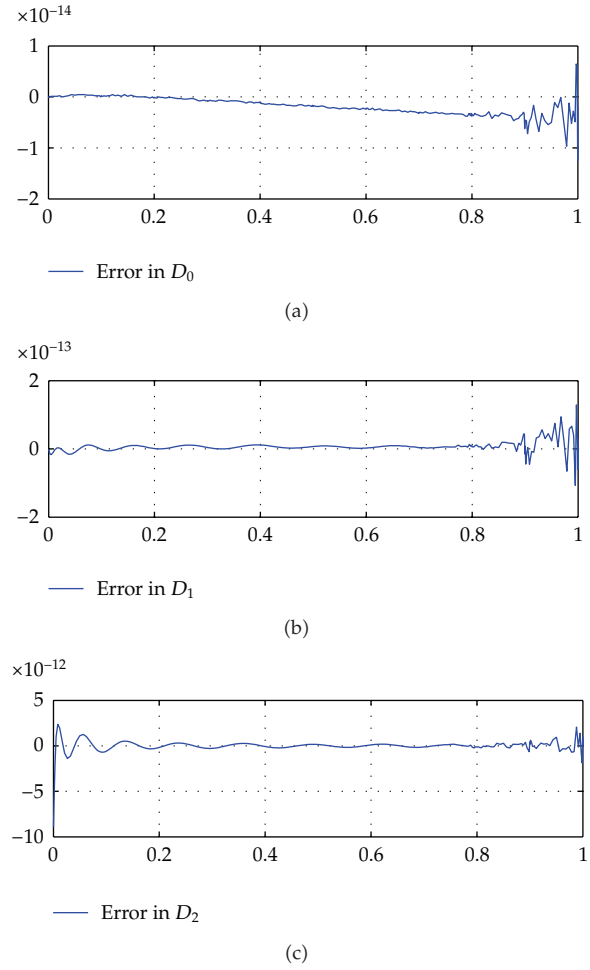

Figure 1: Zero-order Bessel equation: error in the function and derivatives after 200 generations, comparing SIV and analytic solution.

5.2. Linear Interval Transposition: Fourth-Order Differential Equation

Now, it will be considered an example of a fourth-order differential equation. Denoting D_i the i-derivative of the searched function, the equation to be solved is

$$D_4 + D_3 \cdot D_1 - \frac{D_0}{2} + 3 \cdot \sin^2 x + (4D_0 + 3)\cos x + (D_1 - \sin x)^2 + \frac{D_2}{2} = 0, \tag{5.3}$$

$x \in [0, \pi]$, and with the initial condition given by the vector: $[0, 0, 2, 0]$.

Equation (5.3) has analytic solution given by

$$f(x) = x \cdot \sin(x). \tag{5.4}$$

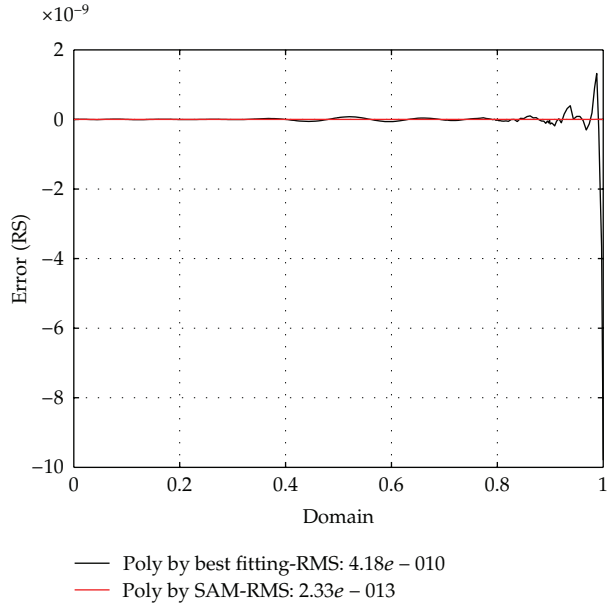

Figure 2: Zero-order Bessel equation: residue due to the best fitting polynomial compared with the residue by SIV polynomial.

In order to transpose the domain from the $[0, \pi]$ to $[-1, 1]$, a linear transformation, as shown in Section 5.1, is used.

Starting with the solution obtained by using MATLAB (ode113) with maximum precision [2], the SIV method with 22 coefficients gives a solution with the error figures shown in Figure 3, after two hundred generations.

To give an idea about the precision of the method, Figure 4 shows the residue by substituting the approximation of the function given by (5.4) by using Legendre's polynomial fitting process [2], and the residue for the same degree evolved solution. Observing Figure 4, it can be noticed that the SIV algorithm improves the accuracy in, at least, two order of magnitude. Besides, the problem of lack of accuracy at the interval extremes is considerably reduced.

5.3. Trigonometric Interval Transposition

Another example where an interval transposition is necessary is described here. Denoting D_i the i-derivative of the searched function, the equation to be solved is

$$D_4 + 4 \cdot D_1 \cdot \frac{\left[3D_1 \cdot x \cdot \left(x^4 - 1\right) - 9 \cdot x^2 + 2\right]}{\left(1 + x^2\right)^3 \cdot \tan^{-1}(x)} = 0, \tag{5.5}$$

$x \in [0, \pi]$, and with the initial condition given by the vector: $[0, 0, 2, 0]$.

Equation (5.3) has analytic solution given by

$$f(x) = \left[\tan^{-1}(x)\right]^2. \tag{5.6}$$

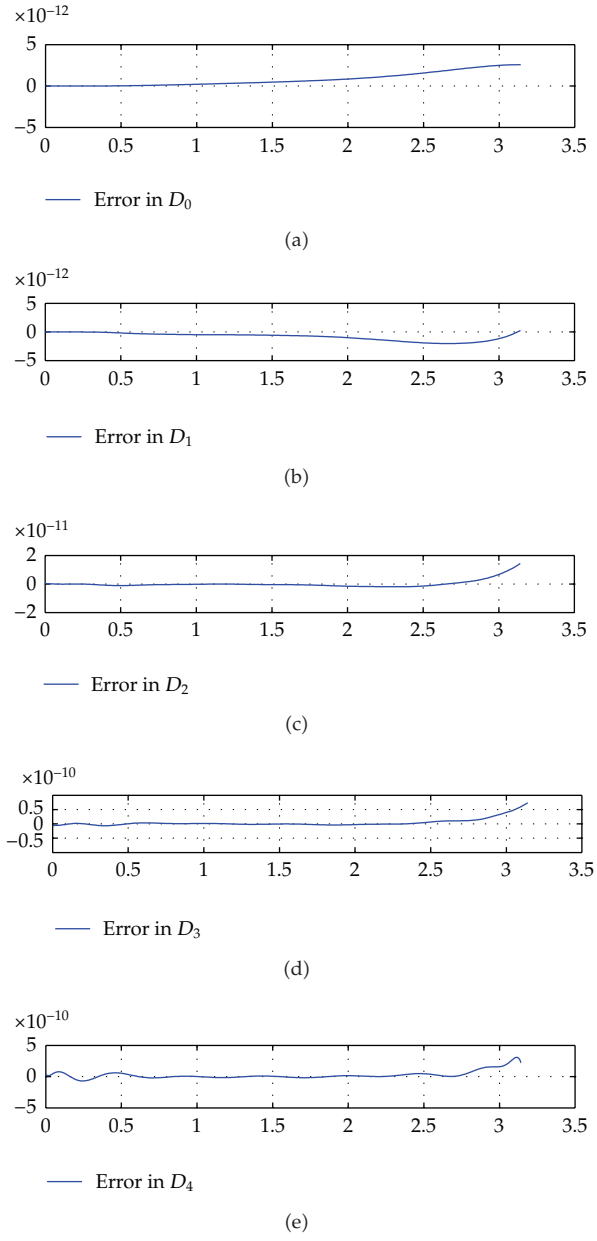

Figure 3: A fourth-order nonlinear equation with linear domain transposition: error in the function and derivatives after 200 generations, comparing SIV and analytic solution.

In order to transpose the domain from $[0, \pi]$ to $[-1, 1]$, a trigonometric transformation is used. For this kind of transformation, considering that a function of x is defined in an interval $[a, b]$, firstly it is transposed with a new function $u = \tan(v)$, defined for an interval $[v_1, v_2]$ of v, such that $v_1 = \tan^{-1}(a)$ and $v_2 = \tan^{-1}(b)$.

Then, this interval is transposed to $[c, d]$ for the variable u, by using the transformation: $u = (d - c) \cdot (\tan^{-1}(x) - v_1)/(v_2 - v_1) + c$.

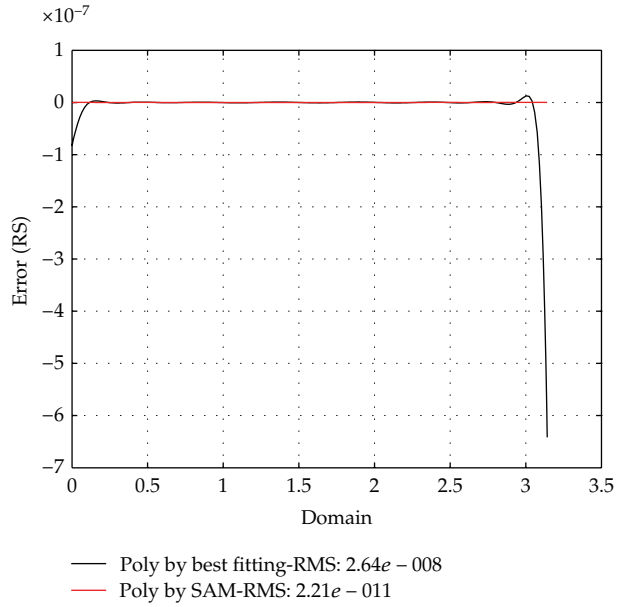

Figure 4: A fourth-order nonlinear equation with linear domain transposition: residue due to the best fitting polynomial compared with the residue due to SIV polynomial.

Starting with the null-polynomial, with seven coefficients and running the SIV process for six hundred generations, taking about 1 minute, a solution with the error figures shown in Figure 5 was obtained.

As a 6th-order approximation for the function given by (5.6) has a very small error, Figure 6 shows the residue of the evolved series, because it is practically the exact solution.

5.4. Equation without Analytic Solution

There are equations such that the analytic solution cannot be expressed by a finite combination of elementary functions. Here, an example of this kind of case is presented. Denoting D_i the i-derivative of the searched function, the equation to be solved is

$$D_3 + D_2 \cdot D_0 = 0, \tag{5.7}$$

$x \in [0, 5]$, and with the initial condition given by the vector $[0,\ 0,\ .4699]$.

After a linear interval transposition, the SIV algorithm started with a 21-degree null-polynomial. After four hundred generations, a solution with residual errors shown in Figure 7 was obtained.

As the analytic solution cannot be expressed as a finite combination of elementary functions, in order to establish comparisons, the Adams' method, implemented with the maximum MATLAB precision [2], was used to perform the Legendre's polynomial fitting for the function. Figure 8 shows that the SIV algorithm improves the precision in, at least, one order of magnitude, attenuating the lack of accuracy at the domain extremes.

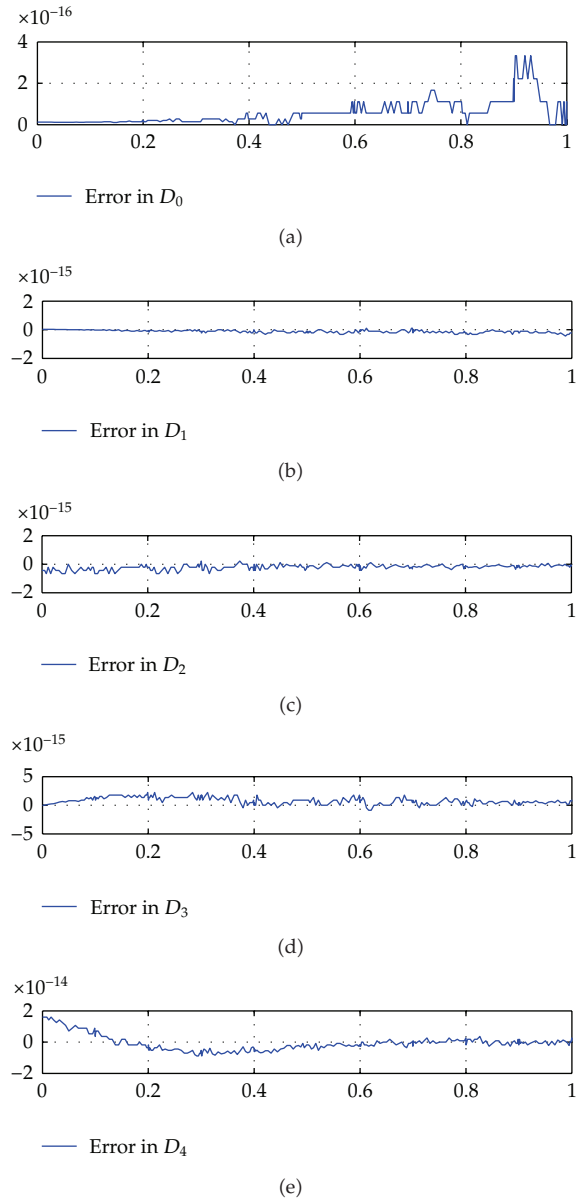

Figure 5: A fourth-order nonlinear equation with trigonometric domain transposition: error in the function and derivatives after 600 generations, comparing SIV and analytic solution.

6. Conclusions

A method that combines genetic algorithms with Legendre's polynomial approximation was proposed, in order to solve nonlinear initial-value problems, up to 4th-order differential equations.

The method presents several advantages.

(i) It provides analytical expressions for the solutions and their derivatives for the whole domain, instead of a low-degree piecewise polynomial;

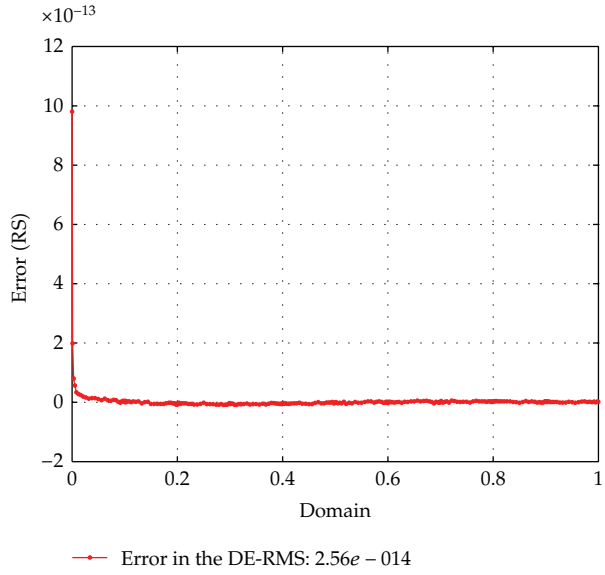

Figure 6: A fourth-order nonlinear equation with trigonometric domain transposition: residue due to SIV polynomial.

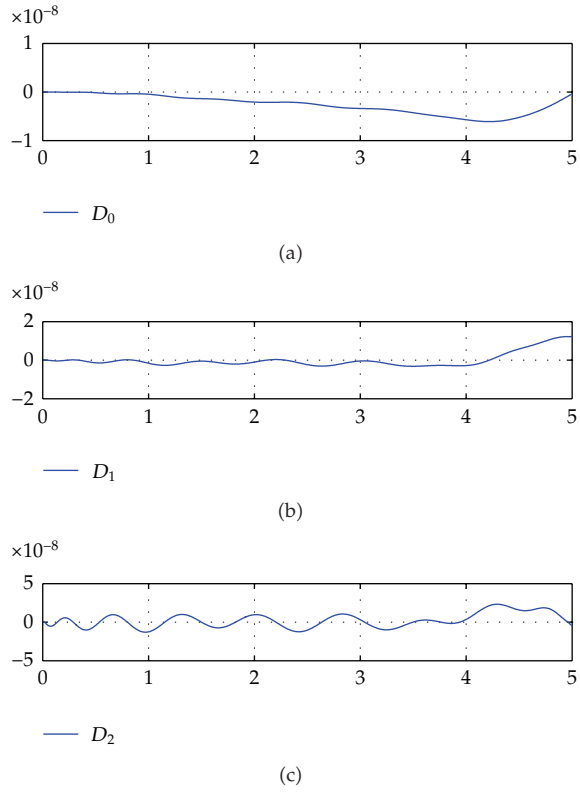

Figure 7: A third-order nonlinear equation without analytic solution: error in the function and derivatives after 600 generations, comparing SIV and ode 113.

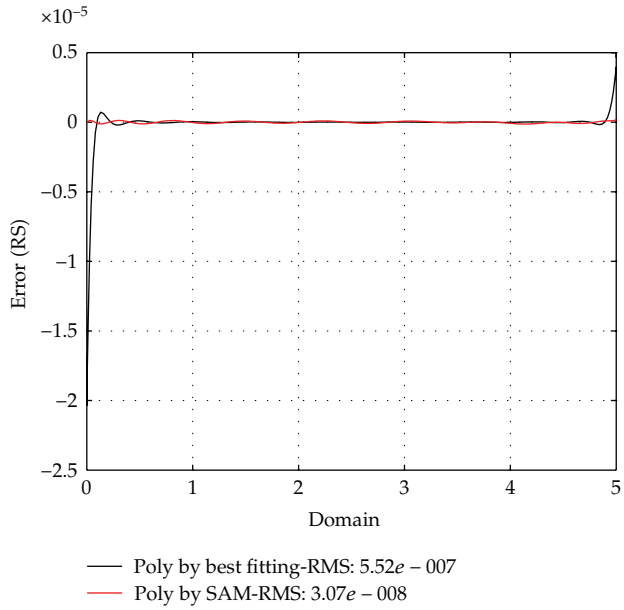

Figure 8: A third-order nonlinear equation without analytic solution: comparing residues in ode 113 and SIV.

(ii) It runs even if the mass matrix is singular, differently from the traditional Runge-Kutta and Adams' methods;

(iii) It is able to start the process of finding solution from a null-polynomial;

(iv) It considerably improves the precision of the solutions, solving the problem related to the lack of accuracy in the interval extremes;

(v) The spectral evolution runs fast once the evolution of each coefficient does not affect the others, due to orthogonality.

Appendices

A. Proofs of the Theorems

Proof of Theorem 3.1. Calling P_n' the derivative of P_n, based on [4, 9], it can be written

(i) $(2n+1)P_n = P'_{n+1} - P'_{n-1}$,

(ii) $P_k(-1) = (-1)^k, k \in N$.

Integrating equation (i) in the real interval $[-1, x]$

$$\int_{-1}^{x} P_n(\xi)d(\xi) = \left[\frac{P_{n+1} - P_{n-1}}{2n+1}\right]_{-1}^{x}, \tag{A.1}$$

and considering (ii), the thesis of the theorem holds for $n \in N^*$. If $n = 0$, $\int_{-1}^{x} P_0(\xi)d\xi = P_1(x) + P_O$. □

Proof of Theorem 3.2. The coefficients of the Legendre's polynomial are given by

$$c_k = \int_{-1}^{1} f(x) \cdot P_k(x) \cdot dx \cdot \frac{(2k+1)}{2}. \tag{A.2}$$

Considering that $f(x) = P_n'(x)$ is a $(n-1)$-degree polynomial and noticing that a n-degree Legendre's polynomial is orthogonal to all polynomials, Legendre's or not, with lesser degree, the integral given by (A.2) can be calculated by parts as

$$\frac{2c_k}{2k+1} = \int_{-1}^{1} \underbrace{P_k}_{u} \cdot \underbrace{P_n' \cdot dx}_{dv} = P_k \cdot P_n|_{-1}^{1} - \int_{-1}^{1} \underbrace{P_n \cdot P_k' \cdot dx}_{\left(\text{orthogonality}\right) = 0}. \tag{A.3}$$

Remembering (ii) from Theorem 3.1 and that $P_k(1) = 1, \quad k \in N$ [4, 9],

$$\frac{2c_k}{2k+1} = 1 - (-1)^k \cdot (-1)^n, \tag{A.4}$$

and, consequently the coefficients of the expansion of P_n' are

$$c_k = \frac{(2k+1)}{2}\left[1 - (-1)^{k+n}\right]. \tag{A.5}$$

□

Proof of Theorem 3.3. As α is a linear operation,

$$\alpha\left[f(u)\right] = \sum_{k=0}^{n} c_k \alpha[B_k(u)]. \tag{A.6}$$

Considering the space of the finite matrices $M_{i,j}$; $i = 1, 2 \ldots n+1$; $j = 1, 2 \ldots n+1$, it can be noticed that, in this space $\alpha[f(u)]$ is an element of the scalar field, consequently, $(\alpha[f(u)])^T = \alpha[f(u)]$.

As B is an $1 \times (n+1)$ vector and $\alpha[B] = ZB$, $VB = C(ZB)$, belonging to a scalar field. Therefore,

$$\begin{aligned} B^T V^T &= (ZB)^T C^T, \\ B^T V^T &= B^T\left(Z^T C^T\right). \end{aligned} \tag{A.7}$$

Taking (A.7) and left multiplying by B results in $V^T = Z^T C^T$. □

B. Detailed Algorithm

The SIV algorithm is implemented divided into three parts: database, adaptive-evolutive process, and restarting, as described below.

B.1. Part 1—Database

(i) Reading problem data: equation, domain, initial conditions, domain transposition type,

(ii) the database is mounted in the transposed domain, according to Section 3.

B.2. Part 2—Adaptive-Evolutive Algorithm

(i) The number of coefficients of expansion is chosen. The program used here is prepared up to 41 coefficients;

(ii) each chromosome (individual) is a set of coefficients of a series expansion of the solution and each gene is a coefficient;

(iii) the zero element of the process is the null polynomial. The coefficients are adjusted following the idea expressed by (3.10) by changing q (order of the equation) coefficients, in order to obey the initial conditions. Consequently, the first winner appears;

(iv) the winner is replaced into (4.1) with G being obtained in several points of the domain. In order to avoid the Runge phenomenon, the chosen points are the nodes of the polynomial basis from the evolutive process;

(v) the series expansion for G is obtained in the chosen basis for the evolutive process by using the Gauss-Legendre numerical integration;

(vi) having the integration matrix and the initial conditions, the series representing G is integrated q times;

(vii) the winner coefficients are compared with the obtained in the former step. If they are equal, the solution is a exact one. If they are not equal, the error bands can be determined for each coefficient, restricting the search space, improving the speed and efficiency of the process;

(viii) having the coefficients error bands, the winner is used as the central element to construct a new population with coefficients varying randomly in the band that works as a standard deviation for each coefficient;

(ix) the population is submitted to a tournament with the ranking depending on the obtained residue substituting the candidate into the original equation. If the winner of the tournament surpasses the former winner, the process returns to the fourth step. If does not, the process goes to the next step;

(x) the individual evaluation in the tournament is performed simply replacing the candidate in the equation and analyzing the residue;

(xi) the lower the residue is, the better is the candidate ranking;

(xii) a mutation is performed in a few genes (about 5%). The 20% worst are discharged, and the champion is preserved for a new round. A new population is formed by the combination of the former population members. The process returns to the 9th step up to the number of generations chosen when the algorithm started to run;

(xiii) the last generation is considered to be the final winner and, after a domain transposition, the results are presented.

B.3. Part 3—Restarting the Process

It is possible to restart the evolution starting with the former results, increasing or maintaining the number of coefficients, beginning in the 8th step of part 2.

References

[1] D. Basmadjian, *Mathematical Modeling of Physical Systems: An Introduction*, Oxford University Press, New York, NY, USA, 2003.

[2] S. Lynch, *Dynamical Systems with Applications Using MATLAB*, Birkhuser, Boston, Mass, USA, 2004.

[3] L. F. Shampine and M. W. Reichelt, "The MATLAB ODE suite," *Society for Industrial and Applied Mathematics Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997.

[4] F. W. Byron, and R. W. Fuller, *Mathematics of Classical and Quantum Physics*, Dover Publications, New York, NY, USA, 1992.

[5] C. L. Karr, I. Yakushin, and K. Nicolosi, "Solving inverse initial-value, boundary-value problems via genetic algorithm," *Engineering Applications of Artificial Intelligence*, vol. 13, no. 6, pp. 625–633, 2000.

[6] P. K. Bergey and C. Ragsdale, "Modified differential evolution: a greedy random strategy for genetic recombination," *Omega*, vol. 33, no. 3, pp. 255–265, 2005.

[7] N. Tutkun, "Parameter estimation in mathematical models using the real coded genetic algorithms," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3342–3345, 2009.

[8] C. B. Moler, *Numerical Computing with MATLAB*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 2004.

[9] G. B. Arfken and H. J. Weber, *Mathematical Methods for Physicists*, Academic Press, San Diego, Claif, USA, 1995.

[10] M. Razzaghi and S. Yousefi, "The Legendre wavelets operational matrix of integration," *International Journal of Systems Science*, vol. 32, no. 4, pp. 495–502, 2001.

[11] O. Guimarães, J. R. C. Piqueira, and M. L. Netto, "Direct computation of operational matrices for polynomial bases," *Mathematical Problems in Engineering*, vol. 2010, Article ID 139198, 12 pages, 2010.

[12] O. Guimarães and J. R. C. Piqueira, *Computação evolutiva na resolução de equaçõesdiferenciais ordinárias não lineares no espaço de Hilbert*, Ph.D. thesis, Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil, 2008.

[13] E. M. E. Elbarbary, "Legendre expansion method for the solution of the second- and fourth-order elliptic equations," *Mathematics and Computers in Simulation*, vol. 59, no. 5, pp. 389–399, 2002.

[14] E. Babolian and F. Fattahzadeh, "Numerical solution of differential equations by using Chebyshev wavelet operational matrix of integration," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 417–426, 2007.

[15] F. B. Liu, "A modified genetic algorithm for solving the inverse heat transfer problem of estimating plan heat source," *International Journal of Heat and Mass Transfer*, vol. 51, no. 15-16, pp. 3745–3752, 2008.

[16] T. K. Sengupta, D. Kalyanmoy,, and B. T. Srikanth, "Control of flow using genetic algorithm for a circular cylinder executing rotary oscillation," *Computers and Fluids*, vol. 36, no. 3, pp. 578–600, 2007.

[17] K. M. Bryden, D. A. Ashlock, S. Corns, and S. J. Willson, "Graph-based evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 550–567, 2006.

[18] H. Q. Cao, L. S. Kang, T. Guo, Y. P. Chen, and H. De Garis, "A two-level hybrid evolutionary algorithm for modeling one-dimensional dynamic systems by higher-order ODE models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 30, no. 2, pp. 351–357, 2000.

[19] D. Moscovitch, O. Noivirt, A. Mezer et al., "Determination of a unique solution to parallel proton transfer reactions using the genetic algorithm," *Biophysical Journal*, vol. 87, no. 1, pp. 47–57, 2004.

[20] M. Sayeed and G. K. Mahinthakumar, "Efficient parallel implementation of hybrid optimization approaches for solving groundwater inverse problems," *Journal of Computing in Civil Engineering*, vol. 19, no. 4, pp. 329–340, 2005.

[21] H. Nakanishi and M. Sugawara, "Numerical solution of the Schrödinger equation by a microgenetic algorithm," *Chemical Physics Letters*, vol. 327, no. 5-6, pp. 429–438, 2000.

[22] P. Chaudhury and S. P. Bhattacharyya, "Numerical solutions of the Schrödinger equation directly or perturbatively by a genetic algorithm: test cases," *Chemical Physics Letters*, vol. 296, no. 1-2, pp. 51–60, 1998.

[23] E. Lutz, "Exact Gaussian quadrature methods for near-singular integrals in the boundary element method," *Engineering Analysis with Boundary Elements*, vol. 9, no. 3, pp. 233–245, 1992.

[24] A. M. Mughal, Y. Xiu, and I. Kamran, "Computational algorithm for higher order legendre polynomial and Gaussian quadrature method," in *Proceedings of the International Conference on Scientific Computing*, pp. 73–77, Las Vegas, Nev, USA, June, 2006.
[25] P. Kaelo and M. M. Ali, "Differential evolution algorithms using hybrid mutation," *Computational Optimization and Applications*, vol. 37, no. 2, pp. 231–246, 2007.