Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2012, Article ID 205391, 25 pages
doi:10.1155/2012/205391

Research Article

Solving Singular Two-Point Boundary Value Problems Using Continuous Genetic Algorithm

Omar Abu Arqub,[1] Zaer Abo-Hammour,[2] Shaher Momani,[3] and Nabil Shawagfeh[3]

[1] *Department of Mathematics, Faculty of Science, Al Balqa Applied University, Salt 19117, Jordan*
[2] *Department of Mechatronics Engineering, Faculty of Engineering, The University of Jordan, Amman 11942, Jordan*
[3] *Department of Mathematics, Faculty of Science, The University of Jordan, Amman 11942, Jordan*

Correspondence should be addressed to Shaher Momani, shahermm@yahoo.com

Received 3 July 2012; Accepted 11 September 2012

Academic Editor: Svatoslav Staněk

In this paper, the continuous genetic algorithm is applied for the solution of singular two-point boundary value problems, where smooth solution curves are used throughout the evolution of the algorithm to obtain the required nodal values. The proposed technique might be considered as a variation of the finite difference method in the sense that each of the derivatives is replaced by an appropriate difference quotient approximation. This novel approach possesses main advantages; it can be applied without any limitation on the nature of the problem, the type of singularity, and the number of mesh points. Numerical examples are included to demonstrate the accuracy, applicability, and generality of the presented technique. The results reveal that the algorithm is very effective, straightforward, and simple.

1. Introduction

Singular boundary value problems (BVPs) for ordinary differential equations arise very frequently in many branches of applied mathematics and physics such as gas dynamics, nuclear physics, chemical reactions, atomic structures, atomic calculations, and study of positive radial solutions of nonlinear elliptic equations (e.g., see [1–3]). In most cases, singular two-point BVPs do not always have solutions which we can obtain using analytical methods. In fact, many of real physical phenomena encountered, are almost impossible to solve by this technique, these problems must be attacked by various approximate and numerical methods.

The purpose of this letter is to introduce the continuous genetic algorithm (CGA), previously developed by the second author, as an alternative to existing methods in solving singular two-point BVPs of the form

$$y''(x) = f(x, y(x), y'(x)), \quad x \in I, \tag{1.1}$$

subject to the boundary conditions

$$y(a) = \alpha, \qquad y(b) = \beta, \tag{1.2}$$

where I is an open or half-open interval with endpoints a and b, a, b, α, β are real finite constants, and f is linear or nonlinear function of y and y'.

When applied to the singular two-point BVP, standard numerical methods designed for regular BVP suffer from a loss of accuracy or may even fail to converge [4], because of the singularity. However, the finite difference method can be used to solve linear singular two-point BVPs, but it can be difficult to solve nonlinear singular two-point BVPs. Furthermore, the finite difference method requires some major modifications that include the use of some root-finding technique while solving nonlinear singular two-point BVPs.

Special numerical methods have been proposed to handle the singular problem. To mention a few, in [5], the author has discussed existence and uniqueness of solutions of singular BVP $y''(x) + (\alpha/x)y'(x) = f(x, y(x))$, including the approximation of solutions via finite difference method. In [6], the author has discussed existence and uniqueness of solutions of singular equation $y''(x) = f(x, y(x), y'(x))$ and presented variable mesh methods for numerically solving such problems. The homotopy analysis method has been applied to solve the singular equation $(1/p(x))y''(x) + (1/q(x))y'(x) + (1/r(x))y(x) = f(x)$ as described in [7]. Furthermore, the higher order finite difference and cubic spline methods are carried out in [8, 9] for the singular BVP $y''(x) + (k/x)y'(x) + q(x)y(x) = f(x)$. In [10] also, the authors have provided the four-order accurate cubic spline method to further investigate the singular equation $y''(x) + (a/x)y'(x) + (a/x^2)y(x) = f(x)$. Also, the reproducing kernel method for solving the singular BVP $y''(x) + (1/p(x))y'(x) + (1/q(x))y(x) = f(x)$ is proposed in [11]. Recently, the modified Adomian decomposition method for solving the singular equation $y''(x) + (1/p(x))y'(x) + (1/q(x))N(y(x)) = f(x)$ is presented in [12].

The reader is kindly requested to go through the survey paper [13] in order to know more details about singular two-point BVPs. In that paper, the authors introduced various numerical techniques including finite difference, splines, finite element, collocation, variational iteration, and other special approximation methods used in literature followed by own critical comments as remarks for solving linear and nonlinear singular problems. However, in most of the present references, the problems discussed are mostly special cases of the general form (1.1) and (1.2), and there are few valid methods of solving (1.1) and (1.2). Hence, one has to go for nonstandard methods.

CGA (The term "continuous" is used to emphasize that the continuous nature of the optimization problem and the continuity of the resulting solution curves) depends on the evolutions of curves in one-dimensional space. The algorithm begins with a population of randomly generated candidates and evolves towards better solution by applying genetic operators which are reproduction, crossover, and mutation. This novel approach is a relatively new class of optimization technique, which generates a growing interest in the mathematics

and engineering community. CGA is well suited for a broad range of problems encountered in science and engineering [14–22].

CGA was developed by the second author [14] as an efficient method for the solution of optimization problems in which the parameters to be optimized are correlated with each other or the smoothness of the solution curve must be achieved. It has been successfully applied in the motion planning of robot manipulators, which is a highly nonlinear, coupled problem [15, 16], in the numerical solution of regular two-point BVPs [17], in the solution of optimal control problems [18], in the solution of collision-free path planning problem for robot manipulators [19], in the numerical solution of Laplace equation [20], and in the numerical solution of nonlinear regular system of second-order BVPs [21]. Their novel development has opened the doors for wide applications of the algorithm in the fields of mathematics and engineering. It has been also applied in the solution of fuzzy differential equations [22]. The reader is asked to refer to [14–22] in order to know more details about CGA, including their justification for use, conditions on smoothness of the functions used in the algorithm, several advantages of CGA over conventional GA (discrete version) when it is applied to problems with coupled parameters and/or smooth solution curves, and so forth.

The work presented in this paper is motivated by the needs for a new numerical technique for the solution of singular two-point BVPs with the following characteristics.

(1) It does not require any modification while switching from the linear to the nonlinear case; as a result, it is of versatile nature.

(2) This approach does not resort to more advanced mathematical tools; that is, the algorithm should be simple to understand, implement, and should be thus easily accepted in the mathematical and engineering application's fields.

(3) The algorithm is of global nature in terms of the solutions obtained as well as its ability to solve other mathematical and engineering problems.

However, being a variant of the finite difference scheme with truncation error of the order $O(h^{10})$, the method provides solutions with moderate accuracy.

The organization of the remainder of this paper is as follows: in the next section, we formulate the singular two-point BVPs. Section 3 covers the description of CGA in detail. Numerical results and discussion are given in Section 4. Finally, concluding remarks are presented in Section 5.

2. Formulation of the Singular Two-Point BVPs

In this section, (1.1) and (1.2) are first formulated as an optimization problem based on the minimization of the cumulative residual of all unknown interior nodes. After that, a fitness function is introduced in order to convert the minimization problem into a maximization problem.

To approximate the solution of (1.1) and (1.2), we make the stipulation that the mesh points are equally distributed through the interval I. This condition is ensured by setting $x_i = a + ih$, $i = 0, 1, \ldots, N$, where $h = (b - a)/N$. Thus, at the interior mesh points, x_i, $i = 1, 2, \ldots, N - 1$, the equation to be approximated is given as

$$F\left(x_i, y(x_i), y'(x_i), y''(x_i)\right) := y''(x_i) - f\left(x_i, y(x_i), y'(x_i)\right) = 0, \quad x_1 \le x_i \le x_{N-1}, \tag{2.1}$$

subject to the boundary conditions

$$y(x_0) = \alpha, \qquad y(x_N) = \beta. \tag{2.2}$$

The difference quotients approximation formulas, which closely approximate $y'(x_i)$ and $y''(x_i)$, $i = 1, 2, \ldots, N-1$, using an $(n+1)$-point at the interior mesh points with error up to $O(h^{n-m+1})$, where $n = 2, 3, \ldots, N$ and $m = 1, 2$ is the order of the derivative can be easily obtained by using Algorithm (6.1) in [23]. For example, based on that algorithm the $(3+1)$-point formulas, with truncation error of order $O(h^3)$, for approximating $y'(x_i)$, are given as

$$\begin{aligned}
y'(x_i) &\approx \frac{1}{h}\left(-\frac{11}{6}y(x_i) + 3y(x_{i+1}) - \frac{3}{2}y(x_{i+2}) + \frac{1}{3}y(x_{i+3})\right), \quad i = 0,\\
y'(x_i) &\approx \frac{1}{h}\left(-\frac{1}{3}y(x_{i-1}) - \frac{1}{2}y(x_i) + y(x_{i+1}) - \frac{1}{6}y(x_{i+2})\right), \quad i = 1, 2, \ldots, N-2,\\
y'(x_i) &\approx \frac{1}{h}\left(\frac{1}{6}y(x_{i-2}) - y(x_{i-1}) + \frac{1}{2}y(x_i) + \frac{1}{3}y(x_{i+1})\right), \quad i = N-1,\\
y'(x_i) &\approx \frac{1}{h}\left(-\frac{1}{3}y(x_{i-3}) + \frac{3}{2}y(x_{i-2}) - 3y(x_{i-1}) + \frac{11}{6}y(x_i)\right), \quad i = N,
\end{aligned} \tag{2.3}$$

while the $(4+1)$-point formulas, with truncation error of order $O(h^3)$, for approximating $y''(x_i)$, are given as

$$\begin{aligned}
y''(x_i) &\approx \frac{1}{h^2}\left(\frac{35}{12}y(x_i) - \frac{26}{3}y(x_{i+1}) + \frac{19}{2}y(x_{i+2}) - \frac{14}{3}y(x_{i+3}) + \frac{11}{12}y(x_{i+4})\right), \quad i = 0,\\
y''(x_i) &\approx \frac{1}{h^2}\left(\frac{11}{12}y(x_{i-1}) - \frac{5}{3}y(x_i) + \frac{1}{2}y(x_{i+1}) + \frac{1}{3}y(x_{i+2}) - \frac{1}{12}y(x_{i+3})\right), \quad i = 1,\\
y''(x_i) &\approx \frac{1}{h^2}\left(-\frac{1}{12}y(x_{i-2}) + \frac{4}{3}y(x_{i-1}) - \frac{5}{2}y(x_i) + \frac{4}{3}y(x_{i+1}) - \frac{1}{12}y(x_{i+2})\right), \quad i = 2, 3, \ldots, N-2,\\
y''(x_i) &\approx \frac{1}{h^2}\left(-\frac{1}{12}y(x_{i-3}) + \frac{1}{3}y(x_{i-2}) + \frac{1}{2}y(x_{i-1}) - \frac{5}{3}y(x_i) + \frac{11}{12}y(x_{i+1})\right), \quad i = N-1,\\
y''(x_i) &\approx \frac{1}{h^2}\left(\frac{11}{12}y(x_{i-4}) - \frac{14}{3}y(x_{i-3}) + \frac{19}{2}y(x_{i-2}) - \frac{26}{3}y(x_{i-1}) + \frac{35}{12}y(x_i)\right), \quad i = N.
\end{aligned} \tag{2.4}$$

However, it is clear that the first and last equations in (2.3) and (2.4) approximate the first and second derivatives of $y(x)$ at the boundary points $x_0 = a$ and $x_N = b$ where the solutions are known. Thus, they are neglected, and only we use the remaining formulas.

We mention here that the number n is starting from 2 and gradually increases up to N. To complete the formulation substituting the approximate formulas of $y'(x_i)$ and $y''(x_i)$, $i = 1, 2, \ldots, N-1$ in (2.1), discretized form of this equation is obtained. The resulting algebraic

equations will be a function of $y(x_{i-(n-1)}), y(x_{i-(n-2)}), \ldots, y(x_{i+(n-1)})$, and $x_i, i = 1, 2, \ldots, N-1$. After that, it is necessary to rewrite the discretized equation in the following form:

$$F\left(x_i, y\left(x_{i-(n-1)}\right), y\left(x_{i-(n-2)}\right), \ldots, y\left(x_{i+(n-1)}\right)\right) \approx 0. \tag{2.5}$$

The residual of the general interior node, $i = 1, 2, \ldots, N-1$, denoted by Res, is defined as

$$\text{Res}(i) = F\left(x_i, y\left(x_{i-(n-1)}\right), y\left(x_{i-(n-2)}\right), \ldots, y\left(x_{i+(n-1)}\right)\right). \tag{2.6}$$

The overall individual residual, Oir, is a function of the residuals of all interior nodes. It may be stated as

$$\text{Oir} = \sqrt{\sum_{i=1}^{N-1} (\text{Res}(i))^2}. \tag{2.7}$$

A mapping of the overall individual residual into a fitness function, Fit, is required in the algorithm in order to convert the minimization problem of Oir into a maximization problem of Fit. A suitable fitness function used in this work is defined as

$$\text{Fit} = \frac{\delta}{\delta + \text{Oir}}, \quad \delta \text{ is a small positive number.} \tag{2.8}$$

The individual fitness is improved if a decrease in the value of the Oir is achieved. The optimal solution of the problem, nodal values, will be achieved when Oir approaches zero and Fit approaches unity.

3. Description of the CGA

In this section, a general review of the GA is presented. After that, a detailed description of the CGA is given. As will be shown later, the efficiency and performance of CGA depend on several factors, including the design of the CGA operators and the settings of the system parameters.

GA is based on principles inspired from the genetic and evolution mechanisms observed in natural systems. Its basic principle is the maintenance of a population of solutions to the problem that evolves towards the global optimum. It is based on the triangle of genetic reproduction, evaluation, and selection [24]. Genetic reproduction is performed by means of two basic genetic operators: crossover and mutation. Evaluation is performed by means of the fitness function that depends on the specific optimization problem. Selection is the mechanism that chooses parent individuals with probability proportional to their relative fitness for the mating process.

The construction of GA for any problem can be separated in five distinct and yet related tasks [24]. First, the genetic representation of potential problem solutions. Second, a method for creating an initial population of solutions. Third, the design of the genetic operators. Fourth, the definition of the fitness function. Fifth, the setting of the system parameters, including the population size, probabilities with which genetic operators are

applied, and so forth. Each of the previous components greatly affects the solution obtained as well as the performance of the GA.

The population-based nature of GA gives it two major advantages over other optimization techniques. First, it identifies the parallel behavior of GA that is realized by a population of simultaneously moving search individuals or candidate solution [24]. Implementation of GA on parallel machines, which significantly reduces the CPU time required, is a major interesting benefit of their implicit parallel nature. Second, information concerning different regions of solution space is passed actively between the individuals by the crossover procedure. This information exchange makes GA an efficient and robust method for optimization, particularly for the optimization of functions of many variables and nonlinear functions. On the other hand, the population-based nature of GA also results in two main drawbacks. First, more memory space is occupied; that is, instead of using one search vector for the solution, N_p search vectors are used, which represent the population size. Second, GA normally suffers from computational burden when applied on sequential machines. This means that the time required for solving certain problem using GA will be relatively large. However, the solution time is a major point when we are interested in real time applications. But if off-line solutions are required for any real-life problem, then our major concern will be the accuracy of the solution rather than the time required for the solution. For real-life problems, the computational time might be reduced to achieve real-time processes utilizing its parallel nature which can be applied on parallel computers or FPGA [18].

The fact that GA uses only objective function information without the need to incorporate highly domain-specific knowledge points to both the simplicity of the approach from one side and its versatility from the other. This means that once a GA is developed to handle a certain problem, it can easily be modified to handle other types of problems by changing the objective function in the existing algorithm. This is why GA is classified as a general-purpose search strategy. The stochastic behavior of GA cannot be ignored as a main part that gives them much of their search efficiency. GA employs random processes to explore a response surface for a specific optimization problem. The advantage of this behavior is the ability to escape local minima without supervision [18, 25].

The use of CGA in problems with coupled parameters and/or smooth curves needs some justification [14, 17]. First, the discrete initialization version of the initial population means that neighbouring parameters might have opposite extreme values that make the probability of valuable information in this population very limited, and correspondingly the fitness will be very low. This problem is overcome by the use of continuous curves that eliminate the possibility of highly oscillating values among the neighbouring parameters and result in a valuable initial population. Second, the traditional crossover operator results in a jump in the value of the parameter in which the crossover point lies while keeping the other parameters the same or exchanged between the two parents. This discontinuity results in a very slow converging process. On the other hand, the CGA results in smooth transition in the parameter values during the crossover process. Third, the conventional version of the mutation process changes only the value of the parameter in which the mutation occurs while it is necessary to make some global mutations which affect a group of neighbouring parameters since either the parameters are coupled with each other or curve should be smooth. To summarize, the operators of the CGA are of global nature and applied at the individual level, while the operators of the traditional GA are of local nature and applied at the parameter level. As a result, the operators of the traditional GA result in a step-function-like jump in the parameter values while those of CGA result in smooth transitions.

However, when using GA in optimization problems, one should pay attention to two points; first, whether the parameters to be optimized are correlated with each other or not. Second, whether there is some restriction on the smoothness of the resulting solution curve or not. In case of uncorrelated parameters or nonsmooth solution curves, the conventional GA will perform well. On the other hand, if the parameters are correlated with each other or smoothness of the solution curve is a must, then the CGA is preferable in this case [14–22]. The steps of CGA used in this work are as follows.

(1) *Initialization*: The initialization function used in the algorithm should be smooth from one side and should satisfy constraint boundary conditions from the other side. Two smooth functions that satisfy the boundary conditions are chosen in this work, which include the modified normal gaussian (MNG) function

$$p_j(i) = r(i) + A\exp\left(-0.5\left(\frac{i-\mu}{\sigma}\right)^2\right)\sin\left(\frac{\pi}{N}i\right), \tag{3.1}$$

and the modified tangent hyperbolic (MTH) function

$$p_j(i) = r(i) + A\tanh\left(\frac{i-\mu}{\sigma}\right)\sin\left(\frac{\pi}{N}i\right), \tag{3.2}$$

for each $i = 1, 2, \ldots, N-1$ and $j = 1, 2, \ldots, N_p$, where $p_j(i)$ is the ith variable value for the jth parent, r is the ramp function of the ith variable value and defined as $r(i) = \alpha + ((\beta - \alpha)/N)i$, N_p is the population size, and μ, σ are random numbers within the range $[1, N-1]$ and $]0, (N-1)/3]$, respectively.

The two initialization functions differ from each other by two main criteria: the convex/concave nature and the possibility of any overshoot/undershoot of the concerned function. The MNG function is either convex or concave within the given range of the independent variable while the MTH function is convex in a subinterval of the independent variable and concave in the remaining interval. The MNG function and MTH function, on the other hand, might result in an overshoot or an undershoot, which might exceed the values of the given boundary conditions at some interior mesh points but not at the boundary point $\{a, b\}$ as will be shown later. The two initialization functions are multiplied by the corrector function, $\sin((\pi/N)i)$, which guarantees that the two functions always satisfy the given boundary conditions.

The choice of A depends on the boundary conditions α and β as follows: A is any random numbers within the range $[-3|\beta - \alpha|, 3|\beta - \alpha|]$ if $\beta - \alpha$ differ from zero, within the range $[-3|\alpha|, 3|\alpha|]$ if $\beta - \alpha$ vanished, and within the range $[-(N-1)/3, (N-1)/3]$ if β and α are both vanished. It is to be noted that for both initialization functions, A specifies the amplitude of the corrector function and σ specifies the degree of dispersion. For small σ the parameter μ specifies the center of the MNG function, while μ specifies the intersection point between the ramp function and the MTH function, which determines the convexity point. The two initialization functions together with the ramp function are shown in Figure 1.

The previously mentioned parameters μ, σ, and A are generated randomly due to the fact that the required solutions are not known for us, and in order to make the initial population as much diverse as we can, randomness should be there to remove any bias

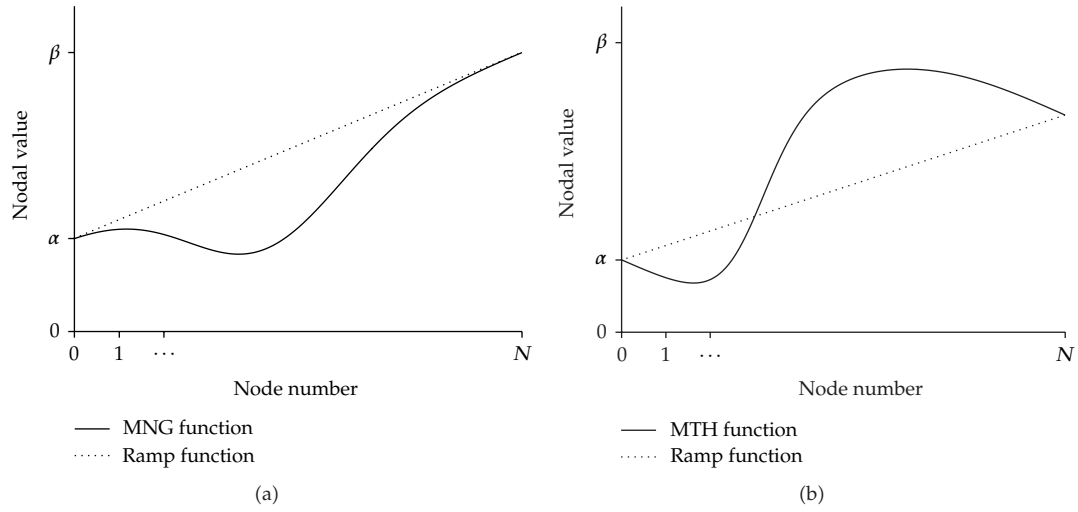


Figure 1: Initialization process.

toward any solution. The mentioned diversity is the key parameter in having an information-rich initial population. In other cases where one of the boundaries of the solution curves is unknown, the reader is kindly requested to go through [18] for comparison and more details.

(2) *Evaluation*: the fitness, a nonnegative measure of quality used to reflect the degree of goodness of the individual, is calculated for each individual in the population.

(3) *Selection*: in the selection process, individuals are chosen from the current population to enter a mating pool devoted to the creation of new individuals for the next generation such that the chance of selection of a given individual for mating is proportional to its relative fitness. This means that the best individuals receive more copies in subsequent generations so that their desirable traits may be passed onto their offspring. This step ensures that the overall quality of the population increases from one generation to the next.

Six selection schemes are incorporated in the algorithm, which include rank-based [26], tournament with replacement [26], tournament without replacement [27], roulette wheel [24], stochastic universal [27], and half-biased selection [28]. Rank-based selection chooses a prescribed number of parent individuals with the highest fitness according to the rank-based ratio, R_{br}, and performs the mating process by choosing parents at random from this subpopulation of the size $R_{br}N_p$.

In the tournament selection scheme, two individuals are randomly selected from the parent population, and a copy of the individual with the large fitness value, better individual, of the two is replaced in the mating pool. Tournament selection has two forms depending on whether the selection individuals will be placed back into the parent population or not. In a tournament without replacement, the two individuals are set aside for the next selection operation, and they are not replaced into the population until all other individuals have also been removed. Since two individuals are removed from the population for every individual selected, the original population is restored after the mating pool is half filled. The process is repeated for a second round in order to fill the mating pool. In a tournament with replacement, upon the selection of the better individual of the two, both individuals are placed back into the original population for the next selection operation. This step is performed until the mating pool is full. When tournament selection schemes are applied,

the number of copies of each individual in the original population cannot be predicted except that it is guaranteed that there will be no copies of the worst individual in the original population.

Roulette wheel selection is a fitness proportionate selection scheme in which the slots of a roulette wheel are sized according to the fitness of each individual in the population. In stochastic universal selection, N_p equidistant markers are placed around the roulette wheel. The number of copies of each individual selected in a single spin of the roulette wheel is equal to the number of markers inside the corresponding slot (the size of slot is still fitness proportional).

Stochastic universal selection guarantees that the number of copies of an individual selected is almost proportional to its fitness, which is not necessarily the case for roulette wheel selection. In half-biased selection, one mate is selected as in roulette wheel selection, while the other mate is selected randomly from the original population.

(4) *Crossover*: crossover provides the means by which valuable information is shared among the individuals in the population. It combines the features of two parent individuals, say s and h, to form two children individuals, say l and $l+1$, that may have new patterns compared to those of their parents and plays a central role in algorithm. The crossover process is expressed as

$$\begin{aligned} c_l(i) &= c(i)p_s(i) + (1 - c(i))p_h(i), \\ c_{l+1}(i) &= (1 - c(i))p_s(i) + c(i)p_h(i), \\ c(i) &= 0.5\left(1 + \tanh\left(\frac{i-\mu}{\sigma}\right)\right), \end{aligned} \tag{3.3}$$

for each $i = 1, 2, \ldots, N-1$, where p_s and p_h represent the two parents chosen from the mating pool, c_l and c_{l+1} are the two children obtained through crossover process, and c represents the crossover weighting function within the range $[0, 1]$. The parameters μ and σ are as given in the initialization process. Figure 2 shows the crossover process in a solution curve for the two random parents. It is clear that new information is incorporated in the children while maintaining the smoothness of the resulting solution curves.

(5) *Mutation*: the mutation function may be any continuous function within the range $[0, 1]$ such that the mutated child solution curve will start with the solution curve of the child produced through the crossover process and gradually changes its value till it reaches the solution curve of the same child at the other end. Mutation is often introduced to guard against premature convergence. Generally, over a period of several generations, the gene pool tends to become more and more homogeneous. The purpose of mutation is to introduce occasional perturbations to the parameters to maintain genetic diversity within the population. The mutation process is governed by the following formulas:

$$\begin{aligned} m_j(i) &= c_j(i) + Am(i), \\ m(i) &= \exp\left(-0.5\left(\frac{i-\mu}{\sigma}\right)^2\right)\sin\left(\frac{\pi}{N}i\right), \end{aligned} \tag{3.4}$$

for each $i = 1, 2, \ldots, N-1$ and $j = 1, 2, \ldots, N_p$, where $c_j(i)$ represents the ith variable value for the jth child produced through the crossover process, $m_j(i)$ is the mutated jth child for

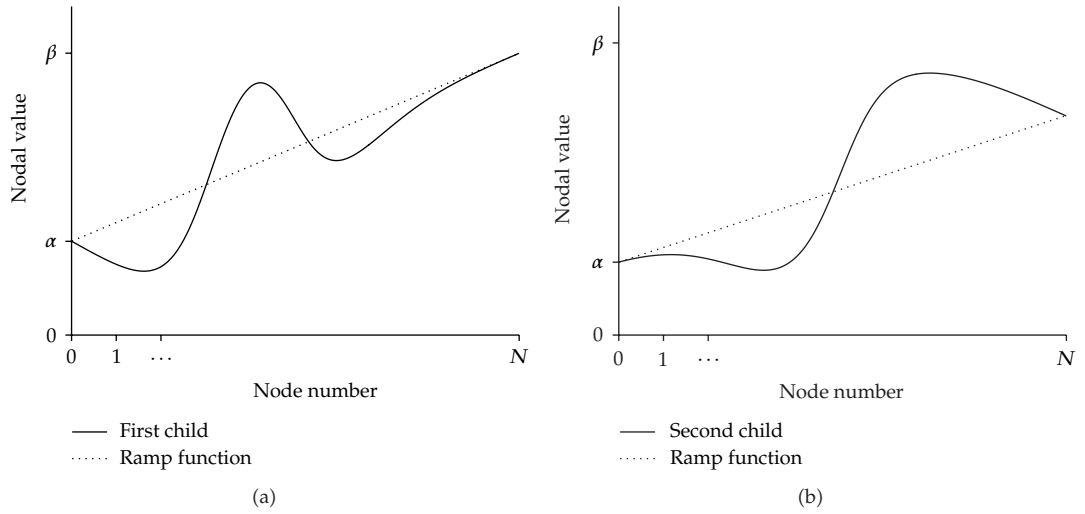


Figure 2: Crossover process.

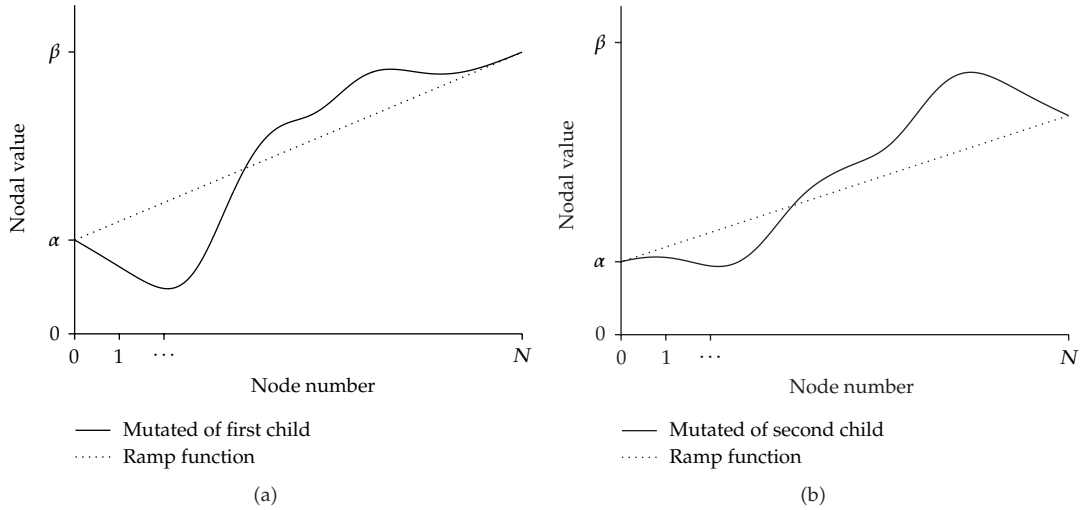


Figure 3: Mutation process.

the ith variable value, and m is the gaussian mutation function. The parameter A is as given in the initialization process.

Regarding the mutation center, μ, and the dispersion factor, σ, used in the mutation process, three methods are used for generating the mutation center where each method is applied to one-third of the population and two methods are used for generating the dispersion factor where each method is applied to one-half of the population. The reader is asked to refer to [17] in order to know more details and descriptions about these methods. The mutation process for a random child is shown in Figure 3. As in the crossover process, some new information is incorporated in the mutated child while maintaining the smoothness of the resulting solution curves.

(6) *Replacement*: after generating the offspring's population through the application of the genetic operators to the parents' population, the parents' population is totally or partially replaced by the offspring's population depending on the replacement scheme used. This is known as nonoverlapping, generational, replacement. This completes the "life cycle" of the population.

(7) *Termination*: the algorithm is terminated when some convergence criterion is met. Possible convergence criteria are as follows: the fitness of the best individual so far found exceeds a threshold value, the maximum nodal residual of the best individual of the population is less than or equals some predefined threshold value, the maximum number of generations is reached, or the improvement in the fitness value of the best member of the population over a specified number of generations is less than some predefined threshold, is reached. After terminating the algorithm, the optimal solution of the problem is the best individual so far found. If the termination conditions are not met, then the algorithm will go back to Step 2.

It is to be noted that the two functions used in the initialization phase of the algorithm will smoothly oscillate between the two ends with a maximum number of single oscillation. If the final solution curves will have more smooth oscillations than one oscillation, then this will be done during the crossover and mutation mechanisms throughout the evolution process. This is actually done by those two operators during the run of the algorithm while solving a problem. However, the evaluation step in the algorithm will automatically decide whether they are rejected or accepted modifications due to their fitness function value.

Two additional operators were introduced to enhance the performance of the CGA, the "elitism" operator, and the "extinction and immigration" operator. These operators are summarized in the form of the following [14–22].

(1) *Elitism*: elitism is utilized to ensure that the fitness of the best candidate solution in the current population must be larger than or equal to that of the previous population.

(2) *Extinction and immigration*: this operator is applied when all individuals in the population are identical or when the improvement in the fitness value of the best individual over a certain number of generations is less than some threshold value. This operator consists of two stages; the first stage is the extinction process where all of the individuals in the current generation are removed except the best-of-generation individual. The second stage is the mass-immigration process where the extinct population is filled out again by generating $N_p - 1$ individuals to keep the population size fixed. The generated population is divided into two equal segments each of $(N_p/2)$ size; the first segment, with $j = 2, 3, 4, \ldots, N_p/2$, is generated as in the initialization phase, while the other segment is generated by performing continuous mutation to the best-of-generation individual as given by the formula

$$p_j(i) = p_1(i) + Am(i), \tag{3.5}$$

for each $i = 1, 2, \ldots, N - 1$ and $j = (N_p/2) + 1, (N_p/2) + 2, \ldots, N_p$, where $p_j(i)$ is the ith variable value for the jth parent generated using immigration operator, p_1 represents the best-of-generation individual, m is the gaussian mutation function, and A represents a random number as given in the initialization process.

To summarize the evolution process in CGA an individual is a candidate solution that consists of 1 curve of $N - 1$ nodal values. The population of individuals undergoes the selection process, which results in a mating pool among which pairs of individuals are crossed over with probability p_c. This process results in an offspring generation where every

Table 1: List of input data to the algorithm.

Parameter	Description
$N_p = 500$	Population size
$p_c = 0.9$	Individual crossover probability
$p_m = 0.9$	Individual mutation probability
$R_{br} = 0.1$	Rank-based ratio
$\delta = 1$	Fitness factor

child undergoes mutation with probability p_m. After that, the next generation is produced according to the replacement strategy applied. The complete process is repeated till the convergence criterion is met where the $N-1$ parameters of the best individual are the required nodal values. The final goal of discovering the required nodal values is translated into finding the fittest individual in genetic terms.

We mention here the following facts about the previously mentioned parameters A, μ, and σ: firstly, the value of these parameters can gradually increase or decrease out of the mentioned intervals that are given in the initialization phase, crossover, and mutation mechanisms throughout the evolution process. Secondly, these values are changed from process to process, from generation to generation, and from curve to curve; this is due to the fact that they are generated randomly.

4. Numerical Results and Discussion

In order to evaluate the performance of the proposed CGA, some problems of singular two-point BVPs are studied. The results obtained by the CGA are compared with the analytical solution of each problem. Results demonstrate that the present method is remarkably effective. The effects of various CGA operators and control parameters on the convergence speed of the proposed algorithm are also investigated in this section. The analysis includes the effect of various initialization methods on the convergence speed of the algorithm in addition to an analysis of the effect of the most commonly used selection schemes, the rank-based ratio, the crossover and mutation probabilities, the population size, the maximum nodal residual, and the step size effect.

The CGA was implemented using visual basic platform. The input data to the algorithm are summarized in Table 1.

Mixed methods for initialization schemes are used where half of the population is generated by the MNG function, while the other half generated using the MTH function. The rank-based selection strategy is used. Generational replacement scheme is applied where the number of elite parents that are passed to the next generation equals one-tenth of the population size. Extinction and immigration operator is applied when the improvement in the fitness value of the best individual of the population over 400 generations is less than 0.001. The termination criterion used for each problem is problem dependent and varies from one case to another. However, the CGA is stopped when one of the following conditions is met.

(1) The fitness of the best individual of the population reaches a value of 0.999999.

(2) The maximum nodal residual of the best individual of the population is less than or equal to 0.00000001.

(3) A maximum number of 3000 generations is reached.

(4) The improvement in the fitness value of the best individual in the population over 500 generations is less than 0.001.

It is to be noted that the first two conditions indicate a successful termination process (optimal solution is found), while the last two conditions point to a partially successful end depending on the fitness of the best individual in the population (near optimal solution is reached) [14–22]. Due to the stochastic nature of CGA, twelve different runs were made for every result obtained in this work using a different random number generator seed; results are the average values of these runs. This means that each run of the CGA will result in a slight different result from the other runs.

Problem 1. Consider the following linear singular two-point BVP with singularity at left endpoint:

$$y''(x) + \frac{1}{x}y'(x) + y(x) = x^2 - x^3 - 9x + 4, \quad 0 < x \leq 1, \tag{4.1}$$

subject to the boundary conditions

$$y(0) = 0, \qquad y(1) = 0. \tag{4.2}$$

The exact solution is $y(x) = x^2 - x^3$.

Problem 2. Consider the following nonlinear singular equation with singularities at both endpoints:

$$\begin{aligned} y''(x) + \frac{60}{\sqrt{x}(x-1)^2}y'(x) + \frac{3}{\tan x}\cos\big(y(x)\big) &= f(x), \quad 0 < x < 1, \\ f(x) &= \frac{3\cos\big(\sin(\pi x) + \exp(1)\big)}{\tan x} - \pi^2 \sin(\pi x) + \frac{60\pi\cos(\pi x)}{\sqrt{x}(x-1)^2}, \end{aligned} \tag{4.3}$$

subject to the boundary conditions

$$y(0) = \exp(1), \qquad y(1) = \exp(1). \tag{4.4}$$

The exact solution is $y(x) = \sin(\pi x) + \exp(1)$.

Problem 3. Consider the following nonlinear singular equation with singularities at both endpoints:

$$\begin{aligned} y''(x) + \frac{1}{x^2(1-x)^3}\frac{y'(x)}{y(x)} + \frac{1}{1-x}y(x) - \frac{f(x)}{y(x)} &= 0, \quad 0 < x < 1, \\ f(x) = \sinh x(\sinh x - x\sinh 1) - \frac{1}{x-1}(\sinh x - x\sinh 1)^2 &- \frac{1}{x^2(x-1)^3}(\cosh x - \sinh 1), \end{aligned} \tag{4.5}$$

Table 2: Convergence data of the four problems.

Problem	Average time (s)	Average generations	Average fitness	Average absolute error	Average absolute residual
1	120.98	889	0.99999900	$3.73551730 \times 10^{-12}$	$5.20529873 \times 10^{-11}$
2	237.39	1227	0.99999097	$2.43196999 \times 10^{-9}$	$3.36319589 \times 10^{-7}$
3	301.45	1024	0.99999900	$2.07980689 \times 10^{-11}$	$5.16836745 \times 10^{-11}$
4	256.70	1202	0.99999715	$1.71191700 \times 10^{-9}$	$3.20372748 \times 10^{-8}$

subject to the boundary conditions

$$y(0) = 0, \qquad y(1) = 0. \tag{4.6}$$

The exact solution is $y(x) = \sinh x - x \sinh 1$.

Problem 4. Consider the following nonlinear singular two-point BVP with singularities at both endpoints:

$$\begin{aligned} y''(x) - \frac{1}{x-1} \tan\left(\frac{1}{\sinh x} y(x) - y'(x)\right) - \cos\left(\frac{1}{x} y(x)\right) &= f(x), \quad 0 < x < 1, \\ f(x) = \frac{1}{x-1} \tan\left(\cosh x + \frac{x}{\sinh x} - 2\right) + \sinh x - \cos\left(1 - \frac{1}{x} \sinh x\right), & \end{aligned} \tag{4.7}$$

subject to the boundary conditions

$$y(0) = 0, \qquad y(1) = \sinh 1 - 1. \tag{4.8}$$

The exact solution is $y(x) = \sinh x - x$.

Throughout this paper, we will try to give the results of the four problems; however, in some cases we will switch between the results obtained for the problems in order not to increase the length of the paper without the loss of generality for the remaining problems and results. The convergence speed of the algorithm, whenever used, means the average number of generations required for convergence. The step size for the four problems is fixed at 0.1, and thus, the number of interior nodes equals 9 for all problems.

The convergence data of the four problems is given in Table 2. It is clear from the table that the problems take about 1086 generations, on average, within about 229.13 seconds to converge to a fitness value of 0.99999653 with an average absolute nodal residual of the value $9.21151501 \times 10^{-8}$ and an average absolute difference between the exact values and the values obtained using CGA of the value $1.04210515 \times 10^{-9}$.

The detailed data of the four problems that includes the exact nodal values, the CGA nodal values, the absolute error, and the absolute nodal residuals is given in Tables 3, 4, 5, and 6, respectively. It is clear that the accuracy obtained using CGA is moderate since it has a truncation error of the order $O(h^{10})$.

Table 3: Numerical results for Problem 1.

Node	Exact value	Approximate value	Absolute error	Absolute residual
0.1	0.009	0.0089999999973	$2.68724626 \times 10^{-12}$	$4.86658042 \times 10^{-11}$
0.2	0.032	0.0319999999954	$4.58882932 \times 10^{-12}$	$4.67374089 \times 10^{-11}$
0.3	0.063	0.0629999999949	$5.07884013 \times 10^{-12}$	$6.04412663 \times 10^{-11}$
0.4	0.096	0.0959999999950	$5.03508346 \times 10^{-12}$	$6.49500377 \times 10^{-11}$
0.5	0.125	0.1249999999952	$4.82547335 \times 10^{-12}$	$4.92386236 \times 10^{-11}$
0.6	0.144	0.1439999999957	$4.28940217 \times 10^{-12}$	$6.36304374 \times 10^{-11}$
0.7	0.147	0.1469999999965	$3.45659612 \times 10^{-12}$	$4.15211688 \times 10^{-11}$
0.8	0.128	0.1279999999976	$2.43857712 \times 10^{-12}$	$4.98899878 \times 10^{-11}$
0.9	0.081	0.0809999999988	$1.21960775 \times 10^{-12}$	$4.34021510 \times 10^{-11}$

Table 4: Numerical results for Problem 2.

Node	Exact value	Approximate value	Absolute error	Absolute residual
0.1	3.0272988228	3.0272988194	$3.44467561 \times 10^{-9}$	$1.63203336 \times 10^{-7}$
0.2	3.3060670808	3.3060670781	$2.62084439 \times 10^{-9}$	$3.96302397 \times 10^{-7}$
0.3	3.5272988228	3.5272988205	$2.34136222 \times 10^{-9}$	$3.23475157 \times 10^{-7}$
0.4	3.6693383448	3.6693383423	$2.43342257 \times 10^{-9}$	$2.94909675 \times 10^{-7}$
0.5	3.7182818285	3.7182818264	$2.09142268 \times 10^{-9}$	$2.71410029 \times 10^{-7}$
0.6	3.6693383448	3.6693383424	$2.32483033 \times 10^{-9}$	$4.97262197 \times 10^{-7}$
0.7	3.5272988228	3.5272988205	$2.34696009 \times 10^{-9}$	$5.84368888 \times 10^{-7}$
0.8	3.3060670808	3.3060670789	$1.86955984 \times 10^{-9}$	$2.97780417 \times 10^{-7}$
0.9	3.0272988228	3.0272988204	$2.41465221 \times 10^{-9}$	$1.98164203 \times 10^{-7}$

Table 5: Numerical results for Problem 3.

Node	Exact value	Approximate value	Absolute error	Absolute residual
0.1	−0.0173533693	−0.0173533694	$1.53966943 \times 10^{-11}$	$6.41444866 \times 10^{-11}$
0.2	−0.0337042362	−0.0337042362	$1.35533459 \times 10^{-11}$	$4.32776073 \times 10^{-11}$
0.3	−0.0480400646	−0.0480400647	$1.76135703 \times 10^{-11}$	$8.11717037 \times 10^{-11}$
0.4	−0.0593281517	−0.0593281517	$2.14053150 \times 10^{-11}$	$5.04949093 \times 10^{-11}$
0.5	−0.0665052913	−0.0665052913	$1.61395758 \times 10^{-11}$	$2.37698996 \times 10^{-11}$
0.6	−0.0684671340	−0.0684671341	$1.93928484 \times 10^{-11}$	$7.63100592 \times 10^{-11}$
0.7	−0.0640571337	−0.0640571337	$2.67523642 \times 10^{-11}$	$4.27937399 \times 10^{-11}$
0.8	−0.0520549727	−0.0520549728	$2.28146529 \times 10^{-11}$	$4.38950218 \times 10^{-11}$
0.9	−0.0311643486	−0.0311643486	$3.41142531 \times 10^{-11}$	$3.92956427 \times 10^{-11}$

Table 6: Numerical results for Problem 4.

Node	Exact value	Approximate value	Absolute error	Absolute residual
0.1	0.0001667500	0.0001667507	$7.18826040 \times 10^{-10}$	$1.71306636 \times 10^{-9}$
0.2	0.0013360025	0.0013360038	$1.20977586 \times 10^{-9}$	$3.19077233 \times 10^{-8}$
0.3	0.0045202934	0.0045202956	$2.10596858 \times 10^{-9}$	$4.13413455 \times 10^{-8}$
0.4	0.0107523258	0.0107523284	$2.60542731 \times 10^{-9}$	$3.19655322 \times 10^{-8}$
0.5	0.0210953055	0.0210953083	$2.78381052 \times 10^{-9}$	$5.57818672 \times 10^{-8}$
0.6	0.0366535821	0.0366535847	$2.50558776 \times 10^{-9}$	$2.54131965 \times 10^{-8}$
0.7	0.0585837018	0.0585837039	$2.02570172 \times 10^{-9}$	$5.20041317 \times 10^{-8}$
0.8	0.0881059822	0.0881059833	$1.14684390 \times 10^{-9}$	$4.70201649 \times 10^{-8}$
0.9	0.1265167257	0.1265167260	$3.05311332 \times 10^{-10}$	$1.18844601 \times 10^{-9}$

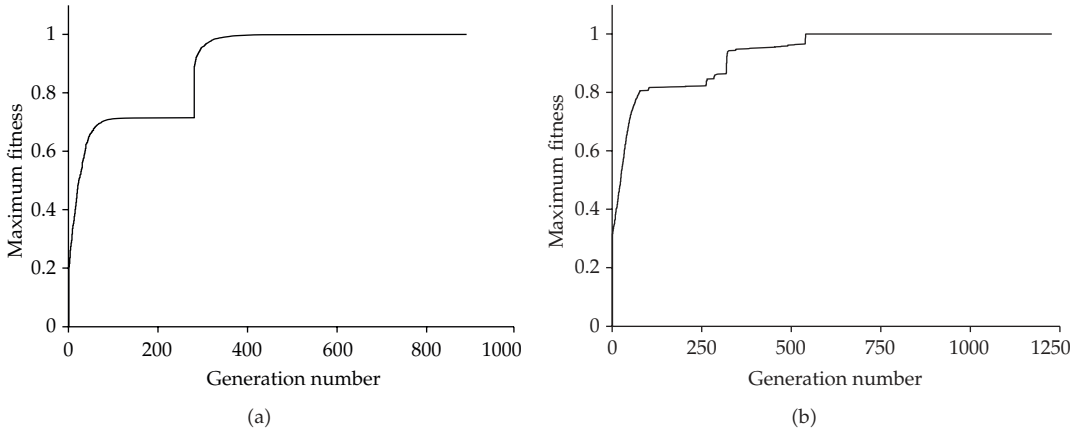


Figure 4: Evolutionary progress plots for the best-of-generation individual for (a) Problem 1; (b) Problem 2.

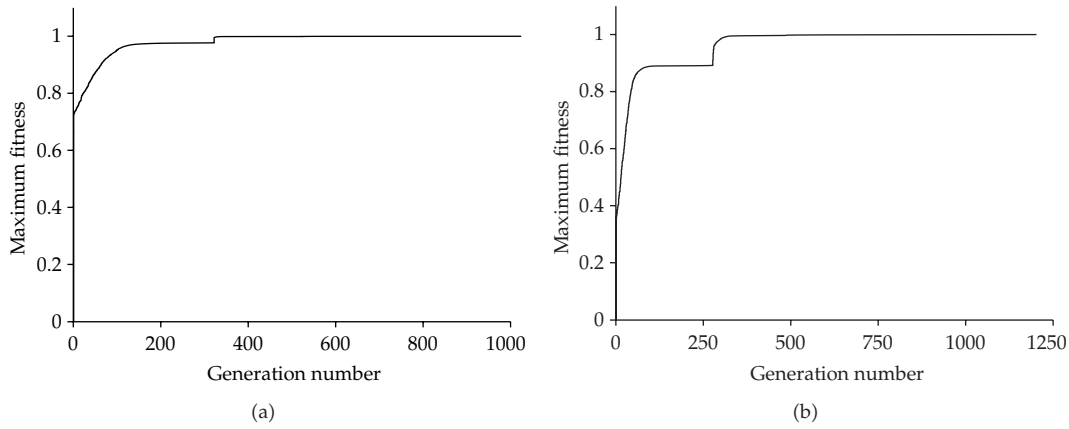


Figure 5: Evolutionary progress plots for the best-of-generation individual for (a) Problem 3; (b) Problem 4.

The evolutionary progress plots of the best-fitness individual of the four problems are shown in Figures 4 and 5. It is clear from the figures that in the first 35% of generations the best fitness approaches to one very fast; after that, it approaches to one slower. That means the approximate of CGA converges to the actual solution very fast in the first 35% of the generations.

The way in which the nodal values evolve for Problems 1 and 4 is studied next. Figure 6 shows the evolution of the first, x_1, middle, x_5, and ninth, x_9, nodal values for Problem 1 while Figure 7 shows the evolution of the second, x_2, middle, x_5, and eighth, x_8, nodal values for Problem 4. It is observed that from the evolutionary plots that the convergence process is divided into two stages: the coarse-tuning stage and the fine-tuning stage, where the coarse-tuning stage is the initial stage in which oscillations in the evolutionary plots occur, while the fine-tuning stage is the final stage in which the evolutionary plots reach steady-state values and do not have oscillations by usual inspection. In other words, evolution has initial oscillatory nature for all nodes, in the same problem. As a

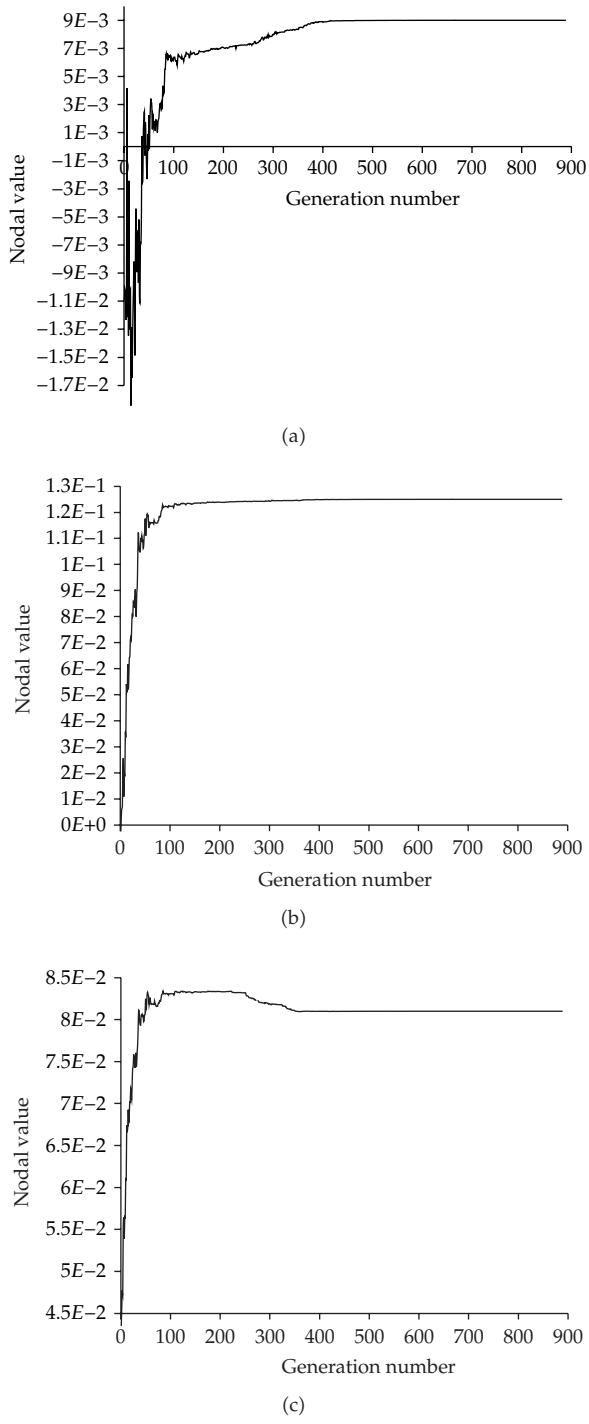
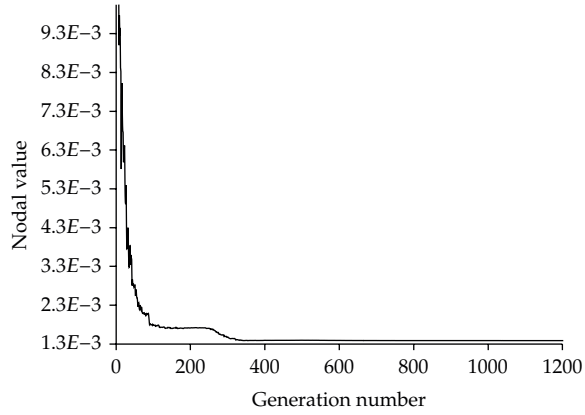
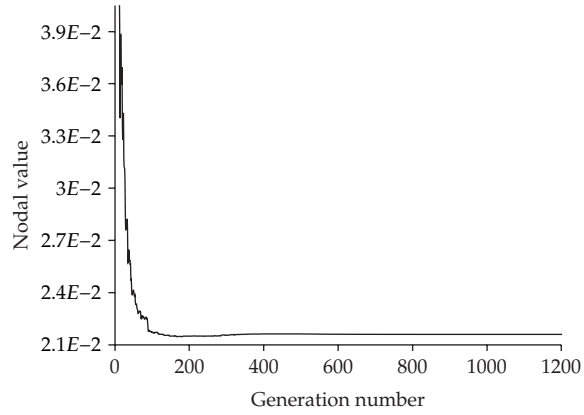


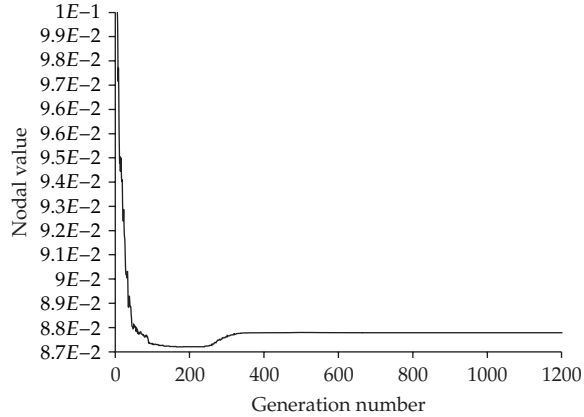
Figure 6: Evolution of the nodal values for Problem 1 of (a) the first node; (b) the middle node; (c) the ninth node.



(a)



(b)



(c)

Figure 7: Evolution of the nodal values for Problem 4 of (a) the second node; (b) the middle node; (c) the eighth node.

Table 7: Average percentage of the coarse-tuning stage of the four problems.

Problem 1	Problem 2	Problem 3	Problem 4
35%	29%	32%	24%

Table 8: Convergence speed of the CGA using different initialization functions.

Initialization method	Problem 1	Problem 2	Problem 3	Problem 4
MNG function	845	1107	916	1372
MTH function	976	1315	1098	1287
Mixed-type functions	889	1227	1024	1202

result, all nodes, in the same problem, reach the near optimal solution together. The average percentage of the fine-tuning stage till convergence from the total number of generations across all nodes of the four problems is given in Table 7. It is clear from the table that the problems spent about 30% of generations, on average, in the coarse-tuning stage, while the remaining 70% is spent in the fine-tuning stage.

The effect of the different types of initialization methods on the convergence speed of the algorithm is discussed next. Three initialization methods are investigated in this work; the first method uses the MNG function, the second uses the MTH function, while the third is the mixed-type initialization method that initializes the first half of the population using the MNG function and the second half of the population using the MTH function. Table 8 shows that the used initialization method has a minor effect on the convergence speed because usually the effect of the initial population dies after few tens of generations and the convergence speed after that is governed by the selection mechanism, crossover, and mutation operators. For Problems 1, 2, and 3, the MNG function results in the fastest convergence speed while for Problem 4, the mixed-type initialization method results in the fastest convergence speed. For a specific problem, the initialization method with the highest convergence speed is the one that provides initial solution curves which are close to the optimal solution of that problem; that is, the optimal solution of the Problems 1, 2, and 3 is close to the MNG function and so on. However, since the optimal solution of any given problem is not assumed to be known, it is better to have a diverse initial population by the use of the mixed-type initialization method. As a result, the mixed-type initialization method is used as the algorithm default method [14–22].

The effect of the most commonly used selection schemes by GA community of the performance on the CGA is explored next. Table 9 represents the convergence speed using the six selection schemes previously described. It is clear from the table that the rank-based selection scheme has the faster convergence speed for all problems. The tournament selection (with and without replacement) approaches come in the second place with almost similar convergence speeds. It is obvious that the fitness proportionate methods (i.e., roulette wheel, stochastic universal, and half-biased selection schemes) have slower convergence speed of the rest of the methods. The half-biased selection scheme has the slowest convergence speed.

The effect of the rank-based ratio, R_{br}, on the convergence speed is studied next. Table 10 gives the convergence speed of the algorithm for different R_{br} value within the range $[0.1, 1]$. It is clear that $R_{br} = 0.1$ results in the best convergence speed for all problems. Furthermore, it is observed that the average number of generations required for convergence increases as the ratio increases.

Table 9: Convergence speed of the CGA using different selection schemes.

Selection method	Problem 1	Problem 2	Problem 3	Problem 4
Rank-based	889	1227	1024	1202
Tournament with replacement	928	1301	1190	1204
Tournament without replacement	945	1312	1204	1209
Roulette wheel	1167	1407	1655	1284
Stochastic universal	1260	1336	1519	1376
Half biased	1311	1641	1720	1810

Table 10: The effect of the rank-based ratio on the convergence speed of the CGA.

R_{br}	Problem 1	Problem 2	Problem 3	Problem 4
0.1	889	1227	1024	1202
0.2	910	1266	1037	1221
0.3	919	1308	1056	1271
0.4	927	1311	1092	1349
0.5	934	1341	1118	1357
0.6	996	1429	1177	1371
0.7	976	1431	1207	1395
0.8	997	1551	1255	1428
0.9	1029	1498	1283	1483
1.0	1048	1562	1342	1536

The effect of the vector norm used in the fitness evaluation is studied here. Two vector norms are used: L_1 norm and L_2 norm. L_1 norm is governed by the following equation:

$$\text{Oir} = \sum_{i=1}^{N-1} |\text{Res}(i)|, \tag{4.9}$$

while L_2 norm is governed by (2.7). Figure 8 shows the evolutionary progress plots for the best-of-generation individual for Problems 2 and 3 using L_1 and L_2 norms while Table 11 gives the convergence speed for the four problems. Two observations are made in this regard; first, the evolutionary progress plots of both norms show that L_2 norm has higher fitness values than those of L_1 norm throughout the evolution process. Second, L_2 norm converges a little bit faster than L_1 norm. The key factor behind these observations is the square power appearing in L_2 norm. Regarding the first observation, it is known that for a given set of nodal residuals with values less than 1, L_1 norm results in a higher value than L_2 norm, and correspondingly, the fitness value using L_2 norm will be higher than that using L_1 norm. Regarding the second observation, L_2 norm tries to select individual solutions, vectors, with distributed nodal residuals among the nodes rather than lumped nodal residuals where one nodal residual is high and the remaining nodal residuals are relatively small. This distributed selection scheme results in closer solutions to the optimal one than the lumped selection scheme. In addition to that, the crossover operator will be more effective in the former case than the latter one. These two points result in the faster convergence speed in L_2 norm as compared with L_1 norm. Furthermore, it is observed that L_2 norm is less sensitive to

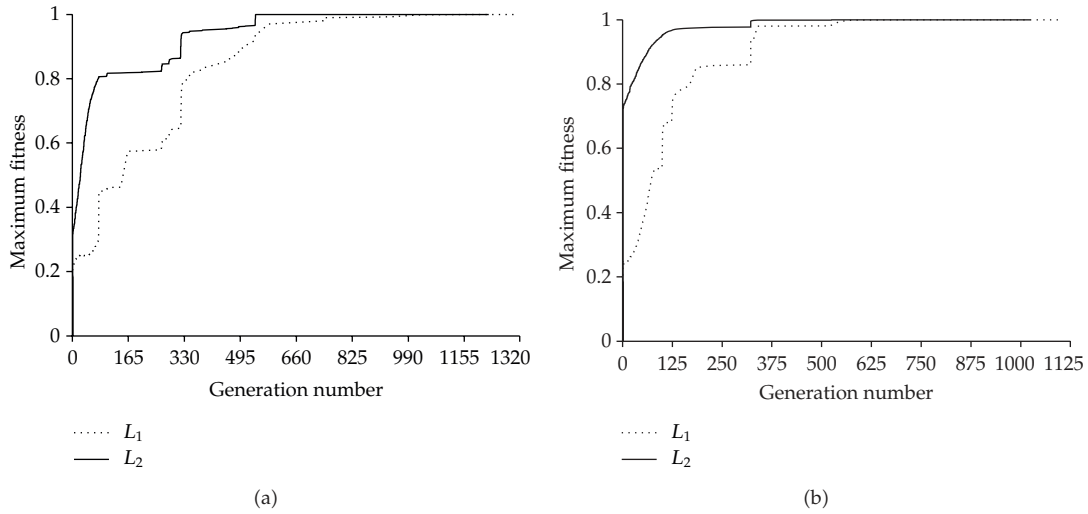


Figure 8: Evolutionary progress plots for the best-of-generation individual using L_1 and L_2 vector norms for (a) Problem 2; (b) Problem 3.

Table 11: The effect of the vector norm on the convergence speed of the CGA.

Vector norm	Problem 1	Problem 2	Problem 3	Problem 4
L_1	968	1305	1096	1287
L_2	889	1227	1024	1202

variations in the genetic related parameters and problem related parameters. As a result, L_2 norm is preferred over L_1 norm, and it is used as the algorithm's default norm [14–22].

The particular settings of several CGA tuning parameters including the probabilities of applying crossover operator and mutation operator are investigated here. These tuning parameters are typically problem dependent and have to be determined experimentally. They play a nonnegligible role in the improvement of the efficiency of the algorithm. Table 12 shows the effect of the crossover probability, p_c, and the mutation probability, p_m, on the convergence speed of the algorithm for Problem 1. The probability value is increased in steps of 0.2 starting with 0.1 and ending with 0.9 for both p_c and p_m. It is clear from the tables that when the probabilities values p_c and p_m are increasing gradually, the average number of generation required for convergence is decreasing as well. It is noted that the best performance of the algorithm is achieved for $p_c = 0.9$ and $p_m = 0.9$. As a result, these values are set as the algorithm default values.

The influence of the population size on the convergence speed of CGA is studied next for Problem 2 as shown in Table 13. The population size is increased in steps of 100 starting with 100 and ending with 1000. Small population sizes suffers from larger number of generations required for convergence and the probability of being trapped in local minima, while large population size suffers from larger number of fitness evaluations that means larger execution time. However, it is noted that the improvement in the convergence speed becomes almost negligible (saturation is reached) after a population size of 600.

Now, the influence of the maximum nodal residual of the best individual on the convergence speed and the corresponding error is investigated. This is the second

Table 12: The effect of the crossover probability and the mutation probability on the convergence speed of the CGA for Problem 1.

(p_m, p_c)	0.1	0.3	0.5	0.7	0.9
0.1	2439	1192	967	930	913
0.3	2360	1188	943	924	912
0.5	2272	1131	925	918	903
0.7	2181	1102	914	907	895
0.9	2089	1080	912	902	889

Table 13: The effect of the population size on the convergence speed and the corresponding error for Problem 2.

N_p	Average generations	Average fitness	Average absolute error	Average absolute residual
100	1988	0.99964220	$3.12718761 \times 10^{-6}$	$3.97612490 \times 10^{-4}$
200	1790	0.99992624	$1.47420585 \times 10^{-7}$	$8.19555830 \times 10^{-5}$
300	1576	0.99995238	$9.93440660 \times 10^{-8}$	$5.29086025 \times 10^{-6}$
400	1401	0.99998115	$4.25985666 \times 10^{-8}$	$2.09453067 \times 10^{-6}$
500	1227	0.99999097	$2.43196999 \times 10^{-9}$	$3.36319589 \times 10^{-7}$
600	1157	0.99999471	$8.92741387 \times 10^{-10}$	$4.43209606 \times 10^{-8}$
700	1127	0.99999684	$3.07785722 \times 10^{-10}$	$2.09929902 \times 10^{-8}$
800	1105	0.99999879	$1.85693238 \times 10^{-10}$	$1.41213808 \times 10^{-8}$
900	1097	0.99999895	$1.00214321 \times 10^{-10}$	$9.94155486 \times 10^{-9}$
1000	1085	0.99999900	$8.95207632 \times 10^{-11}$	$1.44505115 \times 10^{-9}$

termination condition of the algorithm and its value is set between 0.1 and 0.0000000001. Table 14 gives the relevant data for Problem 3. Regarding the convergence speed, it is obvious that as the maximum nodal residual decreases, the number of generations required for convergence increases rapidly since the searching process will be dominated by the fine-tuning stage. The difference between the exact and the CGA nodal values decreases initially till a maximum nodal residual of the value 0.0000000001 is reached. After that, there will be no improvement in the accuracy of the solution obtained for further reduction in the maximum nodal residual. The proposed approach is a variant of the finite difference scheme with a truncation error of order $O(h^{10})$. As a result, the accuracy of the solution obtained is dependent on the step size used, and for a certain step size there will be initial improvement while decreasing the maximum nodal residual till the step size limit is reached where further reduction will be of no use.

The effect of the number of nodes on the convergence speed and the corresponding error is explored next. Table 15 gives the relevant data for Problem 4, where the number of nodes covers the range from 5 to 20 in steps of 5. It is observed that the reduction in the step size results in a reduction in the error and correspondingly an improvement in the accuracy of the obtained solution. This goes in agreement with the known fact about finite difference schemes where more accurate solutions are achieved using a reduction in the step size. On the other hand, the cost to be paid while going in this direction is the rapid increase in the number of generations required for convergence. For instance, while increasing the number of nodes from 5 to 10 to 20, the required number of generations for convergence jumps from almost 550 to 1200 to 3500, that is, 2.15 to 2.51 multiplication factor.

Table 14: The influence of the maximum nodal residual on the convergence speed and the corresponding error for Problem 3.

Max. nodal residual	Average generations	Average fitness	Average absolute error	Average absolute residual
0.1	116	0.73073861	$9.53552997 \times 10^{-4}$	$2.99179323 \times 10^{-2}$
0.01	194	0.98410064	$2.45355430 \times 10^{-6}$	$1.76659532 \times 10^{-4}$
0.001	355	0.99169030	$1.70435356 \times 10^{-7}$	$9.23300479 \times 10^{-5}$
0.0001	657	0.99970196	$2.09730489 \times 10^{-8}$	$6.31153089 \times 10^{-7}$
0.00001	809	0.99996292	$2.64875409 \times 10^{-9}$	$3.47078673 \times 10^{-8}$
0.000001	916	0.99999804	$2.53687869 \times 10^{-10}$	$5.95772308 \times 10^{-9}$
0.0000001	1022	0.99999899	$2.35367806 \times 10^{-11}$	$5.27732298 \times 10^{-11}$
0.00000001	1143	0.99999924	$1.40500158 \times 10^{-11}$	$3.79932887 \times 10^{-11}$
0.000000001	1181	0.99999946	$8.50714082 \times 10^{-12}$	$2.83187520 \times 10^{-11}$
0.0000000001	1195	0.99999951	$5.71884849 \times 10^{-12}$	$8.92419002 \times 10^{-12}$

Table 15: The influence of the number of nodes on the convergence speed and the corresponding error for Problem 4.

Number of nodes	Average generations	Average fitness	Average absolute error	Average absolute residual
5	560	0.99999898	$2.33561226 \times 10^{-7}$	$3.89151263 \times 10^{-6}$
10	1202	0.99999715	$1.71191700 \times 10^{-9}$	$3.20372748 \times 10^{-8}$
20	3017	0.99999672	$1.93030697 \times 10^{-11}$	$4.67261146 \times 10^{-10}$

Table 16: Numerical comparisons of approximate solution for Problem 1.

Node	Exact solution	CGA solution	Method in [11]	Method in [12]
0	0	0	0	0
0.08	0.0058880	0.0058880	0.0058857	0.0058880
0.16	0.0215040	0.0215040	0.0214924	0.0215040
0.32	0.0696320	0.0696320	0.0695762	0.0696320
0.48	0.1198080	0.1198080	0.1196890	0.1198081
0.64	0.1474560	0.1474560	0.1472880	0.1474566
0.80	0.1280000	0.1280000	0.1278430	0.1280032
0.96	0.0368640	0.0368640	0.0368160	0.0368766
1	0	0	0	0.0000170

Finally, numerical comparisons for Problem 1 are studied next. Table 16 shows a comparison between the CGA solution besides the solutions of reproducing kernel Hilbert space method [11] and modified Adomian decomposition method [12] together with the exact solution. As it is evident from the comparison results, it was found that our method in comparison with the mentioned methods is better with a view to accuracy and utilization.

5. Conclusion

In this work, a new numerical algorithm to tackle the singular two-point BVPs is proposed. Central to the approach is the novel use of CGA where smooth solution curves are used for

representing the required nodal values. The proposed technique might be considered as a variation of the finite difference method in the sense that each of the derivatives is replaced by an appropriate difference quotient approximation. The proposed methodology possesses several advantages: first, its ability to solve singular two-point BVPs without the use of other numerical techniques. Second, it does not resort to more advanced mathematical tools; as a result, the present method is found to be simple, efficient, and attractive method with a great potential in mathematical and engineering applications. Third, it does not require any modification while switching from the linear to the nonlinear case. The influence of different parameters, including the evolution of nodal values, the initialization method, the selection method, the rank-based ratio, the vector norm used, the crossover and mutation probabilities, the population size, the maximum nodal residual, and the step size, is also studied. This new method promises to open new possibilities for applications in an important class of mathematical and engineering problems.

References

[1] H. Çağlar, N. Çağlar, and M. Özer, "B-spline solution of non-linear singular boundary value problems arising in physiology," *Chaos, Solitons and Fractals*, vol. 39, no. 3, pp. 1232–1237, 2009.

[2] P. Hiltman and P. Lory, "On oxygen diffusion in a spherical cell with Michaelis-Menten oxygen uptake kinetics," *Bulletin of Mathematical Biology*, vol. 45, no. 5, pp. 661–664, 1983.

[3] S. Chandrasekhar, *Hydrodynamic and Hydromagnetic Stability*, Dover, New York, NY, USA, 1981.

[4] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, vol. 13 of *Classics in Applied Mathematics*, SIAM, Philadelphia, Pa, USA, 1995.

[5] P. Jamet, "On the convergence of finite-difference approximations to one-dimensional singular boundary-value problems," *Numerische Mathematik*, vol. 14, pp. 355–378, 1969/1970.

[6] R. K. Mohanty, "A family of variable mesh methods for the estimates of (du/dr) and solution of non-linear two-point boundary-value problems with singularity," *Journal of Computational and Applied Mathematics*, vol. 182, no. 1, pp. 173–187, 2005.

[7] A. S. Bataineh, M. S. M. Noorani, and I. Hashim, "Approximate solutions of singular two-point BVPs by modified homotopy analysis method," *Physics Letters A*, vol. 372, pp. 4062–4066, 2008.

[8] A. S. V. Ravi Kanth and Y. N. Reddy, "Higher order finite difference method for a class of singular boundary value problems," *Applied Mathematics and Computation*, vol. 155, no. 1, pp. 249–258, 2004.

[9] A. S. V. Ravi Kanth and Y. N. Reddy, "Cubic spline for a class of singular two-point boundary value problems," *Applied Mathematics and Computation*, vol. 170, no. 2, pp. 733–740, 2005.

[10] R. K. Mohanty, P. L. Sachdev, and N. Jha, "An $O(h^4)$ accurate cubic spline TAGE method for nonlinear singular two point boundary value problems," *Applied Mathematics and Computation*, vol. 158, no. 3, pp. 853–868, 2004.

[11] M. Cui and F. Geng, "Solving singular two-point boundary value problem in reproducing kernel space," *Journal of Computational and Applied Mathematics*, vol. 205, no. 1, pp. 6–15, 2007.

[12] C. Chun, A. Ebaid, M. Y. Lee, and E. Aly, "An approach for solving singular two-point boundary value problems: analytical and numerical treatment," *ANZIAM Journal*, vol. 53, pp. E21–E43, 2011.

[13] M. Kumar and N. Singh, "A collection of computational techniques for solving singular boundary-value problems," *Advances in Engineering Software*, vol. 40, no. 4, pp. 288–297, 2009.

[14] Z. S. Abo-Hammour, *Advanced continuous genetic algorithms and their applications in the motion planning of robotic manipulators and the numerical solution of boundary value problems [Ph.D. thesis]*, Quiad-Azam University, 2002.

[15] Z. S. Abo-Hammour, N. M. Mirza, S. M. Mirza, and M. Arif, "Cartesian path generation of robot manipulators using continuous genetic algorithms," *Robotics and Autonomous Systems*, vol. 41, no. 4, pp. 179–223, 2002.

[16] Z. S. Abo-Hammour, "A novel continuous genetic algorithms for the solution of the cartesian path generation problem of robot manipulators," in *Robot Manipulators: New Research*, J. X. Lui, Ed., pp. 133–190, Nova Science, New York, NY, USA, 2005.

[17] Z. S. Abo-Hammour, M. Yusuf, N. M. Mirza, S. M. Mirza, M. Arif, and J. Khurshid, "Numerical solution of second-order, two-point boundary value problems using continuous genetic algorithms," *International Journal for Numerical Methods in Engineering*, vol. 61, no. 8, pp. 1219–1242, 2004.

[18] Z. S. Abo-Hammour, A. G. Asasfeh, A. M. Al-Smadi, and O. M. K. Alsmadi, "A novel continuous genetic algorithm for the solution of optimal control problems," *Optimal Control Applications and Methods*, vol. 32, no. 4, pp. 414–432, 2011.

[19] Z. S. Abo-Hammour, O. Alsmadi, S. I. Bataineh, M. A. Al-Omari, and N. Affach, "Continuous genetic algorithms for collision-free cartesian path planning of robot manipulators," *International Journal of Advanced Robotic Systems*, vol. 8, pp. 14–36, 2011.

[20] Z. S. Abo-Hammour, R. B. Albadarneh, and M. S. Saraireh, "Solution of Laplace equation using continuous genetic algorithms," *Kuwait Journal of Science & Engineering*, vol. 37, no. 2, pp. 1–15, 2010.

[21] O. Abu Arqub, Z. Abo-Hammour, and S. Momani, "Application of continuous genetic algorithm for nonlinear system of second-order boundary value problems," *Applied Mathematics and Information Sciences*. In press.

[22] O. Abu Arqub, *Numerical solution of fuzzy differential equation using continuous genetic algorithms [Ph. D. thesis]*, University of Jordan, 2008.

[23] J. Li, "General explicit difference formulas for numerical differentiation," *Journal of Computational and Applied Mathematics*, vol. 183, no. 1, pp. 29–52, 2005.

[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[25] U. Hörchner and J. H. Kalivas, "Further investigation on a comparative study of simulated annealing and genetic algorithm for wavelength selection," *Analytica Chimica Acta*, vol. 311, no. 1, pp. 1–13, 1995.

[26] J. H. Jiang, J. H. Wang, X. Chu, and R. Q. Yu, "Clustering data using a modified integer genetic algorithm (IGA)," *Analytica Chimica Acta*, vol. 354, no. 1–3, pp. 263–274, 1997.

[27] E. M. Rudnick, J. H. Patel, G. S. Greenstein, and T. M. Niermann, "A genetic algorithm framework for test generation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 9, pp. 1034–1044, 1997.

[28] J. R. Vanzandt, "A genetic algorithm for search route planning," ESD TR-92-262, United States Air Force, 1992.