Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080129

Application of Continuous Genetic Algorithm for Nonlinear System of Second-Order Boundary Value Problems

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

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

Received: 1 Jun. 2012, Revised: 9 Sep. 2012, Accepted: 10 Sep. 2012
Published online: 1 Jan. 2014

Abstract: In this paper, a numerical algorithm, based on the use of genetic algorithm technique, is presented for solving a class of nonlinear systems of second-order boundary value problems. In this technique, the system is formulated as an optimization problem by the direct minimization of the overall individual residual error subject to the given constraints boundary condtions, and is then solved using continuous genetic algorithm. In general, the proposed technique uses smooth operators and avoids sharp jumps in the parameter values. The applicability, efficiency, and accuracy of the proposed algorithm for the solution of different problems is investigated. Meanwhile, the convergence analysis based on the resulting statistical data is also discussed.

Keywords: Continuous genetic algorithm, second-order boundary value problems, finite difference method

1 Introduction

Systems of second-order boundary value problems (BVPs) occur frequently in applied mathematics, engineering, theoretical physics, biology and so on [1,2,3]. If the systems of second-order BVPs cannot be solved analytically because, generally, the solution cannot be exhibited in a closed form even when it exists, which is the usual case, then recourse must be made to numerical and approximate methods.

Many classical numerical methods used with second-order initial value problems cannot be applied to second-order BVPs. We all know that the finite difference method can be used to solve linear second-order BVPs, but it can be difficult to solve nonlinear second-order BVPs. Furthermore, the finite difference method requires some major modifications that include the use of some root-finding technique while solving nonlinear second-order BVPs. For a nonlinear system of second-order BVPs, there are few valid methods to obtain numerical solutions.

In this paper, we apply the continuous genetic algorithm (CGA) (The term ”continuous” is used to emphasize that the continuous nature of the optimization problem and the continuity of the resulting solution curves) for the solution of the following nonlinear system of second-order BVPs [4]:

$$\begin{aligned} & a_{1,0}(x)\,u_1''(x) \\ & +a_{1,1}(x)\,u_1'(x)+a_{1,2}(x)\,u_1(x) \\ & +a_{1,3}(x)\,u_2''(x)+a_{1,4}(x)\,u_2'(x) \\ & +a_{1,5}(x)\,u_2(x)+G_1(x,u_1(x),u_2(x))=f_1(x)\,, \\ & a_{2,0}(x)\,u_2''(x) \\ & +a_{2,1}(x)\,u_2'(x)+a_{2,2}(x)\,u_2(x) \\ & +a_{2,3}(x)\,u_1''(x)+a_{2,4}(x)\,u_1'(x) \\ & +a_{2,5}(x)\,u_1(x)+G_2(x,u_1(x),u_2(x))=f_2(x)\,, \end{aligned} \tag{1}$$

subject to the boundary conditions

$$\begin{aligned} u_1(a)=\alpha_1, u_1(b)=\beta_1, \\ u_2(a)=\alpha_2, u_2(b)=\beta_2, \end{aligned} \tag{2}$$

where $a \le x \le b$, α_k, β_k, $k = 1,2$ are real finite constants, G_1, G_2 are nonlinear functions of u_1, u_2, and f_1, f_2 and $a_{1,i}, a_{2,i}$, $i = 0,1,2,3,4,5$ are continuous functions on $[a,b]$.

The previous studies for system (1) and (2) can be summarized as follows: in [5], the authors have discussed

[*] Corresponding author e-mail: o.abuarqub@bau.edu.jo, s.momani@ju.edu.jo

the existence of solutions, including the approximation of solutions via finite difference equations. Recently, the iterative reproducing kernel method and a combination of homotopy perturbation and reproducing kernel methods are carried out in [6,7]. The authors in [4,8,9] have developed the sinc-collocation, the homotopy perturbation, and the cubic B-spline scaling function methods to solve system (1) and (2). In [10] also, the author has provided the variational iteration method to further investigation to the aforementioned system. Furthermore, the homotopy analysis method has been applied to solve system (1) and (2) as described in [11]. Finally, the B-spline method for solving the linear form of system (1) is proposed recently in [12]. On the other hand, the numerical solvability of other version of differential equations and other related equations can be found in [13,14,15] and references therein.

CGA was developed in [16] 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 [17], in the numerical solution of two-point BVPs [18], and in the solution of optimal control problems [19]. 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 [20]. The reader is asked to refer to [16,17,18] 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, etc.

In this paper, we introduce a novel method based on CGA for numerically approximating a solution of nonlinear systems of second-order BVPs (1) and (2). The new method has the following characteristics; first, it should not require any modification while switching from the linear to the nonlinear case; as a result, it is of versatile nature. Second, its ability to solve nonlinear system (1) and (2) without the use of other numerical techniques. Third, it should 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. Fourth, the algorithm is of global nature in terms of the solutions obtained as well as its ability to solve other mathematical and engineering problems. Fifth, it requires the minimal amount of information about specific problems; as a result, the discretized form of system (1) and (2) is the only required step that differs from one problem to another one.

This paper is organized in six sections including the introduction. In Section 2, a short preface to optimization is presented. In Section 3, we formulate the system of second-order BVPs as an optimization problem. Section 4 covers the description of GA and CGA in detail. Numerical examples and convergence analysis are presented in Section 5. Finally, in Section 6 some concluding remarks are presented.

2 Preface to optimization

Optimization plays a crucial role in various disciplines in sciences, industry, engineering, and almost in every aspect of the daily life. Optimization problems are encountered, for example, in communication systems, antenna design, applied mathematics, medicine, economic, and so on.

In mathematics, statistics, empirical sciences, computer science, or management science, mathematical optimization is the selection of a best element with regard to some criteria from some set of available alternatives. In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. The generalization of optimization theory and techniques to other formulations comprises a large area of applied mathematics. More generally, optimization includes finding best available values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains. Normally, there is no single answer for any optimization problem, and it is necessary to choose the best solution for a given problem from the multitude of possible solutions. To achieve this, it is necessary to define the objective function.

Optimization problems can be divided into two categories depending on whether the solution is continuous or discrete. An optimization problem with discrete solution is known as a conventional optimization problem, while the continuous version is known as a continuous optimization problem.

3 Formulation of the optimization problem

An optimization problem is the problem of finding the best solution from all feasible solutions. In this section, the system (1) is formulated as an optimization problem based on the minimization of the cumulative residual of all unknown interior nodes.

For the first step of formulation, the independent interval $[a,b]$ is partitioned into N subintervals of equal length h given as $h = (b-a)/N$. The mesh points, nodes, are obtained using the equation $x_i = a + ih$, $i = 0,1,\ldots,N$. Thus, at the interior mesh points, x_i, $i = 1,2,\ldots,N-1$, the system to be approximated is given as:

$$\begin{aligned} &F_1(x_i, u_1(x_i), u_1'(x_i), \\ &u_1''(x_i), u_2(x_i), u_2'(x_i), u_2''(x_i)) = 0, \\ &F_2(x_i, u_1(x_i), u_1'(x_i), \\ &u_1''(x_i), u_2(x_i), u_2'(x_i), u_2''(x_i)) = 0, \end{aligned} \tag{3}$$

subject to the boundary conditions

$$u_1(x_0) = \alpha_1, u_1(x_N) = \beta_1,$$
$$u_2(x_0) = \alpha_2, u_2(x_N) = \beta_2,$$

where $x_1 \le x_i \le x_{N-1}$, $i = 1, 2, \ldots, N-1$, and F_1, F_2 are given by:

$$\begin{aligned}&F_1(x_i, u_1(x_i), u_1'(x_i),\\ &u_1''(x_i), u_2(x_i), u_2'(x_i), u_2''(x_i)) = a_{1,0}(x_i)\, u_1''(x_i)\\ &+a_{1,1}(x_i)\, u_1'(x_i) + a_{1,2}(x_i)\, u_1(x_i)\\ &+a_{1,3}(x_i)\, u_2''(x_i) + a_{1,4}(x_i)\, u_2'(x_i)\\ &+a_{1,5}(x_i)\, u_2(x_i) + G_1(x_i, u_1(x_i), u_2(x_i)) - f_1(x_i),\end{aligned}$$

$$\begin{aligned}&F_2(x_i, u_1(x_i), u_1'(x_i),\\ &u_1''(x_i), u_2(x_i), u_2'(x_i), u_2''(x_i)) = a_{2,0}(x_i)\, u_2''(x_i)\\ &+a_{2,1}(x_i)\, u_2'(x_i) + a_{2,2}(x_i)\, u_2(x_i)\\ &+a_{2,3}(x_i)\, u_1''(x_i) + a_{2,4}(x_i)\, u_1'(x_i)\\ &+a_{2,5}(x_i)\, u_1(x_i) + G_2(x_i, u_1(x_i), u_2(x_i)) - f_2(x_i).\end{aligned}$$

The CGA approach for numerically approximating the solution to system (1) and (2) consists of replacing each derivative in the differential system by a difference quotient, which closely approximates that derivative when h is small. On the other hand, the difference approximation formulas, which closely approximate $u_k'(x_i)$ and $u_k''(x_i)$, $k = 1, 2$, $i = 1, 2, \ldots, N-1$ using $(n+1)$-point at the interior mesh points with error of order $\mathrm{O}(h^{n-m+1})$, 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 [21]. 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 $u_k'(x_i)$ and $u_k''(x_i)$, $k = 1, 2$ in system (3), a discretized form of system (1) is obtained. The resulting algebraic equations will be a discrete function of x_i, $u_k(x_{i-(n-1)})$, $u_k(x_{i-(n-2)})$, ..., and $u_k(x_{i+(n-1)})$, $k = 1, 2$. After that, it is necessary to rewrite the discretized system in the form of the following:

$$\begin{aligned}&F_1(x_i, u_1(x_{i-(n-1)}),\\ &u_1(x_{i-(n-2)}), \ldots, u_1(x_{i+(n-1)}),\\ &u_2(x_{i-(n-1)}), u_2(x_{i-(n-2)}), \ldots, u_2(x_{i+(n-1)})) \approx 0,\end{aligned}$$

$$\begin{aligned}&F_2(x_i, u_1(x_{i-(n-1)}),\\ &u_1(x_{i-(n-2)}), \ldots, u_1(x_{i+(n-1)}),\\ &u_2(x_{i-(n-1)}), u_2(x_{i-(n-2)}), \ldots, u_2(x_{i+(n-1)})) \approx 0.\end{aligned}$$

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

$$\begin{aligned}&\mathrm{Res}_1(i) = F_1(x_i, u_1(x_{i-(n-1)}),\\ &u_1(x_{i-(n-2)}), \ldots, u_1(x_{i+(n-1)}),\\ &u_2(x_{i-(n-1)}), u_2(x_{i-(n-2)}), \ldots, u_2(x_{i+(n-1)})),\end{aligned}$$

$$\begin{aligned}&\mathrm{Res}_2(i) = F_2(x_i, u_1(x_{i-(n-1)}),\\ &u_1(x_{i-(n-2)}), \ldots, u_1(x_{i+(n-1)}),\\ &u_2(x_{i-(n-1)}), u_2(x_{i-(n-2)}), \ldots, u_2(x_{i+(n-1)})).\end{aligned}$$

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

$$\mathrm{Oir} = \sqrt{\sum_{i=1}^{N-1} (\mathrm{Res}_1(i))^2 + (\mathrm{Res}_2(i))^2}.$$

The fitness function, Fit, plays a fundamental rule in optimization techniques (continuous and conventional version) and its applications. This function is required in our work in order to convert the minimization problem into a maximization problem. In fact, we do this to facilitate the calculations and planning graphics. A suitable fitness function used in this work is defined as:

$$\mathrm{Fit} = \frac{1}{1 + \mathrm{Oir}}.$$

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

4 Steps of CGA technique

The following account presents a brief review of the GA. After that, a detailed description of the CGA is given. Moreover, the design of the CGA operators and the settings of the system parameters will be shown later to be the key factors on which the efficiency and performance of CGA rely.

A GA is an optimization technique that mimics biological evolution as a problem-solving strategy. Given a specific problem to solve, the input to the GA is a set of potential solutions to that problem, encoded in some fashion, and a metric called a fitness function that allows each candidate to be quantitatively evaluated. These candidates may be solutions already known to work, with the aim of the GA being to improve them, but more often they are generated at random. The GA then evaluates each candidate according to the fitness function. In a pool of randomly generated candidates, of course, most will not work at all, and these will be deleted. However, purely by chance, a few may hold promise-they may show activity, even if only weak and imperfect activity, toward solving the problem.

These promising candidates are kept and allowed to reproduce. Multiple copies are made of them, but the copies are not perfect; random changes are introduced during the copying process. These digital offspring then go on to the next generation, forming a new pool of candidate solutions, and are subjected to a second round of fitness evaluation. Those candidate solutions which were worsened, or made no better, by the changes to their code are again deleted; but again, purely by chance, the random variations introduced into the population may have improved some individuals, making them into better, more complete or more efficient solutions to the problem at hand. Again these winning individuals are selected and copied over into the next generation with random changes, and the process repeats. The expectation is that the average fitness of the population will increase each round, and so by repeating this process for hundreds or thousands of rounds, very good solutions to the problem can be discovered.

As astonishing and counterintuitive as it may seem to some, GA has proven to be an enormously powerful and successful problem-solving strategy, dramatically demonstrating the power of evolutionary principles. GA has been used in a wide variety of fields to evolve solutions to problems as difficult as or more difficult than those faced by human designers. Moreover, the solutions they come up with are often more efficient, more elegant, or more complex than anything comparable a human engineer would produce.

On the other aspect, GA can be distinguished from calculus-based and enumerative methods for optimization by the following characteristics [22,23,24,25,26,27,28]:

1. GA searches for optimal solution using a population of individuals, not a single individual. This very important characteristic gives GA much of its search power and also points to its parallel nature.

2. GA uses only objective function information. No other auxiliary information is required. Much of the interest in GA technique is due to the fact that they belong to the class of efficient domain-independent search strategies that are usually superior in performance to traditional methods without the need to incorporate highly domain-specific knowledge.

3. GA uses probabilistic transition rules, not deterministic rules, in contrast with the calculus based and enumerative methods.

*Remark.*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 non-smooth 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 [16,17,18,19,20].

The CGA proposed in this work consists of the following steps [16,17,18,19,20]:

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 function (MNGF):

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

and the modified tangent hyperbolic function (MTHF):

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

for each $i = 1,2,\ldots,N-1$, $j = 1,2,\ldots,N_p$, and $k = 1,2$, where $p_j(k,i)$ is the i-th variable value of the k-th curve for the j-th parent, r is the ramp function of the i-th variable value of the k-th curve and defined as:

$$r(k,i) = \alpha_k + \tfrac{\beta_k-\alpha_k}{N}i,$$

μ,σ are random numbers within the range $[1,N-1]$ and $(0,(N-1)/3]$, respectively, and N_p is the population size.

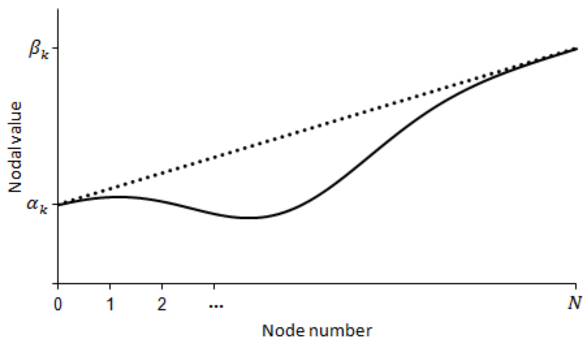
The two initialization functions differ from each other by two main criteria: the convex or concave nature and the possibility of any overshoot or undershoot of the concerned function. The MNGF is either convex or concave within the given range of the independent variable while the MTHF is convex in a subinterval of the independent variable and concave in the remaining interval. The MNGF and MTHF, 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, $\sin(\pi i/N)$, which guarantees that the two functions always satisfy the given boundary conditions.

The choice of A depend on the boundary conditions α_k and β_k, $k = 1,2$ as follows: A is any random number within the range $[-3|\beta_k-\alpha_k|, 3|\beta_k-\alpha_k|]$ if $\beta_k-\alpha_k$ differ from zero, within the range $[-3\alpha_k, 3\alpha_k]$ if $\beta_k-\alpha_k$ vanished, and finally within the range $[-(N-1)/3, (N-1)/3]$ if β_k and α_k 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 MNGF, while μ specifies the intersection point between the ramp function and the MTHF, which determines the convexity point. The two initialization functions together with the ramp function are shown in Figure 1.

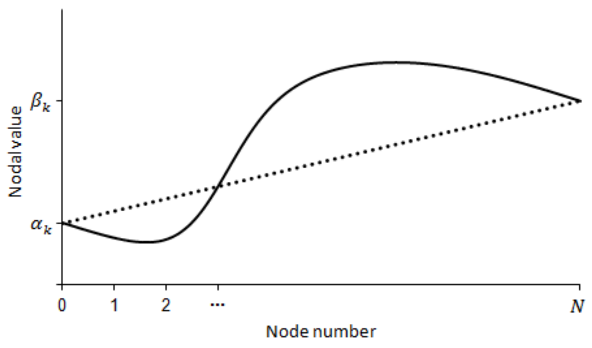
The previously mentioned parameter μ,σ and A are generated randomly due to the fact that the required solutions are not known for us, and in order to make the initial population as much diverse as we can, randomness should be there to remove any bias toward any solution. The mentioned diversity is the 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 [19] 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



(a)



(b)

Fig. 1 Initialization process: (a) ——— MNGF and $\cdots\cdots$ ramp function; (b) ——— MTHF and $\cdots\cdots$ ramp function.

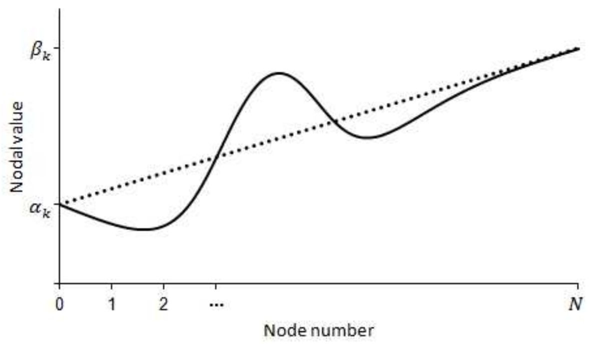
devoted to the creation of new individuals for the next generation such that the chance of selection of a given individual for mating is proportional to its relative fitness. This means that the best individuals receive more copies in subsequent generations so that their desirable traits may be passed onto their offspring. This step ensures that the overall quality of the population increases from one generation to the next.

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:

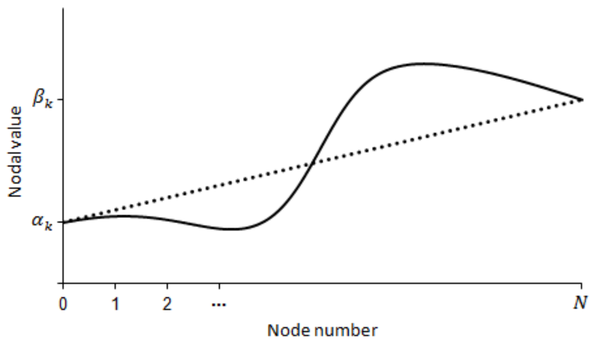
$$c_l\left(k,i\right) = c\left(k,i\right) p_s\left(k,i\right) + \left(1 - c\left(k,i\right)\right) p_h\left(k,i\right),$$
$$c_{l+1}\left(k,i\right) = \left(1 - c\left(k,i\right)\right) p_s\left(k,i\right) + c\left(k,i\right) p_h\left(k,i\right),$$
$$c\left(k,i\right) = 0.5\left(1 + \tanh\left(\tfrac{i-\mu}{\sigma}\right)\right),$$

for each $i = 1,2,\ldots,N-1$ and $k = 1,2$, 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.



(a)



(b)

Fig. 2 Crossover process: (a) ——— first child and $\cdots\cdots$ ramp function; (b) ——— second child and $\cdots\cdots$ ramp function.

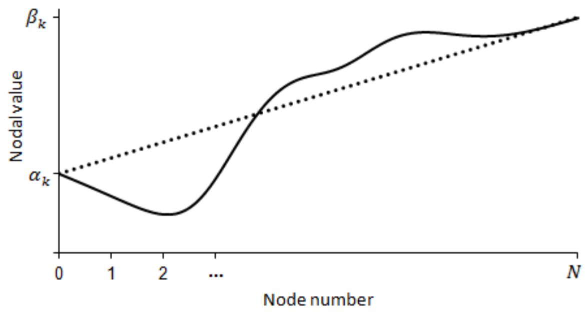
5. **Mutation:** The mutation function may be any continuous function within the range $[0,1]$ such that the mutated child solution curve will start with the solution curve of the child produced through the crossover process and gradually changes its value till it reaches the solution curve of the same 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:

$$m_j\left(k,i\right) = c_j\left(k,i\right) + Am\left(k,i\right)\sin\left(\tfrac{\pi}{N}i\right),$$
$$m\left(k,i\right) = \exp\left(-0.5\left(\tfrac{i-\mu}{\sigma}\right)^2\right),$$

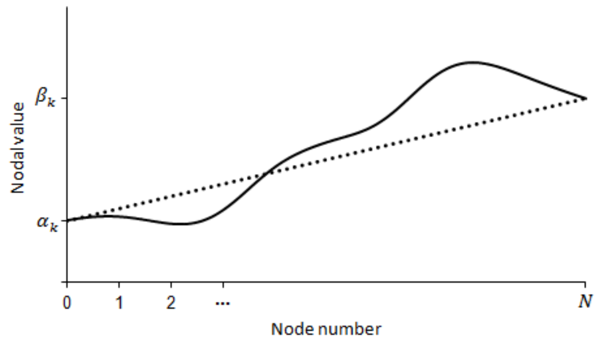
for each $i = 1,2,\ldots,N-1$, $j = 1,2,\ldots,N_p$, and $k = 1,2$, where c_j represents the j-th child produced through the crossover process, m_j is the mutated j-th child, and m is the gaussian mutation function. The parameter A is as given in the initialization process.

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 [18] 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.



(a)



(b)

Fig. 3 Mutation process: (a) ——— mutated of first child and ······ ramp function; (b) ——— mutated of second child and ······ ramp function.

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 non-overlapping, generational, replacement. This completes the "life cycle" of the population.

7. **Termination:** The CGA is terminated when some convergence criterion is met. Possible convergence criteria are: 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.

*Remark.*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 vary from process to process, from generation to generation, and from curve to curve; this is due to the fact that they are generated randomly.

It is to be noted that the two functions used in the initialization phase of the CGA 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.

To summarize the evolution process in CGA an individual is a candidate solution that consists of two curves each 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_{ci} within that pair of parents, individual solution curves are crossed with probability p_{cc}. This process results in an offspring generation where every child undergoes mutation with probability p_{mi}, within that child individual solution curves are mutated with probability p_{mc}. 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 two curves of the best individual are the required solution curves. The final goal of discovering the required nodal values is translated into finding the fittest individual in genetic terms.

5 Numerical and convergence analysis

In this section, some numerical problems are studied to demonstrate the accuracy and applicability of the present technique. Results obtained are compared with the analytical solution of each problem in different ways and are found to be in good agreement with each other. 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 curve crossover and curve mutation probabilities, population size, maximum nodal residual, and the step size effect. In the process of computation, all the symbolic and numerical computations were performed by using Visual Basic platform.

The CGA proposed in this paper is used to solve the given system of second-order BVPs. The input data to the algorithm is divided into two parts; the CGA related parameters and the system of second-order BVPs related parameters. The CGA related parameters include the population size, N_p, the individual crossover probability, p_{ci}, the curve crossover probability, p_{cc}, the individual mutation probability, p_{mi}, the curve mutation probability, p_{mc}, the initialization method, the selection scheme used, the replacement method, the immigration threshold value and the corresponding number of generations, and finally the termination criterion. The system of second-order BVP related parameters include the governing differential system, the independent interval $[a,b]$, the step size, h, the boundary values, α,β, and finally the number of nodes, N. However, the input data to the algorithm are as follows:

Parameter	Description
$N_p = 500$	Population size
$p_{ci} = 0.9$	Individual crossover probability
$p_{cc} = 0.5$	Curve crossover probability
$p_{mi} = 0.9$	Individual mutation probability
$p_{mc} = 0.5$	Curve mutation probability
$R_{br} = 0.1$	Rank-based ratio

In order to the initialization, mixed methods for initialization schemes in the CGA are used where half of the population is generated by the MNGF, while the other half generated using the MTHF. The rank-based selection strategy is used where the rank-based ratio is set to 0.1. 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. The termination criterion used for each problem is a problem dependent and varies from one case to another. However, the CGA is stopped when one of the following conditions is met. First, the fitness of the best individual of the population reaches a value of 0.99999. Second, the maximum nodal residual of the best individual of the population is less than or equal to 0.00000001. Third, a maximum number of 3000 generations is reached. Fourth, 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 to 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) [16,17,18,19,20].

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. On the other hand, the system of second-order BVPs related parameters are depend on the nature of the problem and will be determined later in the same problem.

Problem 1. Consider the following linear system [8]:

$$u_1''(x) + xu_1(x) + xu_2(x) = f_1(x),$$

$$u_2''(x) + 2xu_1(x) + 2xu_2(x) = f_2(x),$$

subject to the boundary conditions

$$u_1(0) = 0, u_1(1) = 0,$$

$$u_2(0) = 0, u_2(1) = 0,$$

where $0 \le x \le 1$, $f_1(x) = 2$, and $f_2(x) = -2$. The exact solutions are: $u_1(x) = x^2 - x$ and $u_2(x) = x - x^2$.

Problem 2. Consider the following nonlinear system [4]:

$$u_1''(x) + xu_1(x) + 2xu_2(x) + xu_1^2(x) = f_1(x),$$

$$u_2'(x) + u_2(x) + x^2u_1(x) + \sin(x)\,u_2^2(x) = f_2(x),$$

subject to the boundary conditions

$$u_1(0) = 0, u_1(1) = 0,$$

$$u_2(0) = 0, u_2(1) = 0,$$

where $0 \le x \le 1$, $f_1(x) = 2x\sin(\pi x) + x^2 - 2x^4 + x^5 - 2$, and $f_2(x) = \sin(\pi x)(1 + \sin(x)\sin(\pi x)) + \pi\cos(\pi x) + x^3 - x^4$. The exact solutions are: $u_1(x) = x - x^2$ and $u_2(x) = \sin(\pi x)$.

Problem 3. Consider the following nonlinear system [7]:

$$u_1''(x) + 20u_1'(x)$$
$$+4\cos(x)\,u_1(x) + \sin(u_1(x)\,u_2(x)) = f_1(x),$$

$$u_2''(x) + 5e^x u_2'(x)$$
$$+6\sinh(x)\,u_2(x) + \cos(u_2(x)) = f_2(x),$$

subject to the boundary conditions

$$u_1(0) = 1, u_1(1) = e,$$

$$u_2(0) = 0, u_2(1) = \sinh(1),$$

where $0 \le x \le 1$, $f_1(x) = \sin(e^x\sinh(x)) + 21e^x + 4e^x\cos(x)$, and $f_2(x) = \cos(\sinh(x)) + 11\cosh^2(x) + 5\sinh(x)\cosh(x) + \sinh(x) - 6$. The exact solutions are: $u_1(x) = e^x$ and $u_2(x) = \sinh(x)$.

Throughout this paper, we will try to give the results of the three 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 three problems is fixed at 0.1 and thus, the number of interior nodes equals 9 for all problems.

The convergence data of the three problems is given in Table 1. It is clear from the table that the problems take 1387 generations, on average, to converge to a fitness value of about 0.99999469 with an average absolute nodal residual of the value $3.31605833 \times 10^{-7}$ and an average absolute nodal error of the value $2.87790635 \times 10^{-8}$.

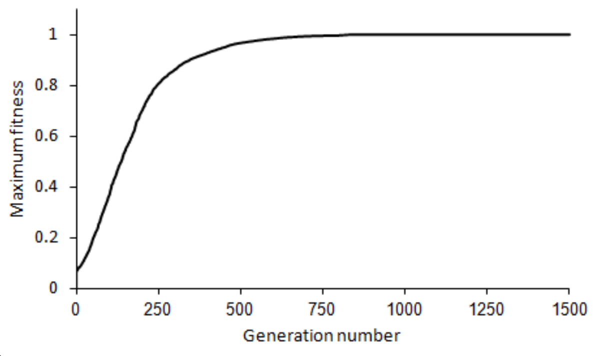
The evolutionary progress plots, of the best-fitness individual of Problems 1, 2, and 3 are shown in Figure 4. It is clear from the figure that, in the first 50% of generations the best-fitness approaches to one very fast, after that, it approaches to one slower. That means the approximate of CGA converge to the actual solution very fast in the first 50% of the generations.

The way in which the nodal values evolve for Problem 3 is studied next. Figure 5 shows the evolution of the first, x_1, and middle, x_5, nodal values of $u_1(x)$, while Figure 6 shows the evolution of the middle, x_5, and ninth, x_9, nodal values of $u_2(x)$.

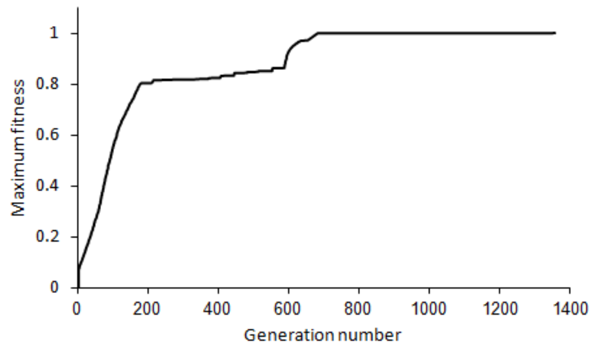
It is observed that from the evolutionary plots that the convergence process is divided into two stages: the coarse-tuning stage and the fine-tuning stage, where the coarse-tuning stage is the initial stage in which oscillations in the evolutionary plots occur, while the fine-tuning stage is the final 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 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 three problems is given in Table 2. 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 graphs of the exact and approximate solutions of $u_1(x)$ and $u_2(x)$ for Problem 1 are depicted in Figure 7, while the graphs of the absolute error and absolute residual across all nodes are plotted in Figure 8. It is to be noted that the accuracy of certain node is inversely proportional to its distance (number of nodes) from the boundaries. From the last mentioned Figures, we see that we can achieve a good approximation with the exact solution.

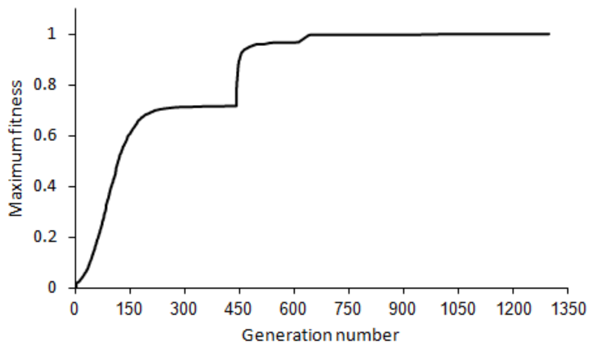
The effect of the step size on the convergence speed and the corresponding maximum error and maximum residual is explored next. Tables 3 and 4 give the relevant data for Problem 2, where the number of nodes covers the range from 10 to 80. It is observed that the reduction in the step size results in 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 reducing the step size from 0.1 to 0.05, the required number of generations for convergence jumps from almost 1300 to 1600, i.e. 1.23 multiplication factor.



(a)



(b)



(c)

Fig. 4 Evolutionary progress plots for the best-of-generation individual across all generations for: (a) Problem 1; (b) Problem 2; (c) Problem 3.

Table 1 Convergence data of the three problems.

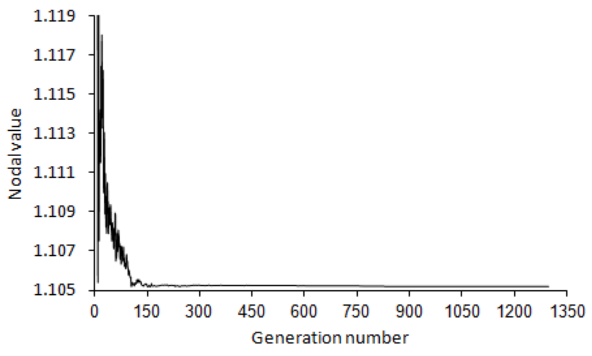
Problem	Average generations	Average fitness	Average error	Average residual
1	1503	0.99999765	$3.74820144 \times 10^{-8}$	$5.29666249 \times 10^{-7}$
2	1359	0.99999303	$4.64375608 \times 10^{-8}$	$3.87627057 \times 10^{-7}$
3	1299	0.99999339	$2.41761536 \times 10^{-9}$	$7.75241930 \times 10^{-8}$

Table 2 Average percentage of the coarse-tuning stage of the three problems.

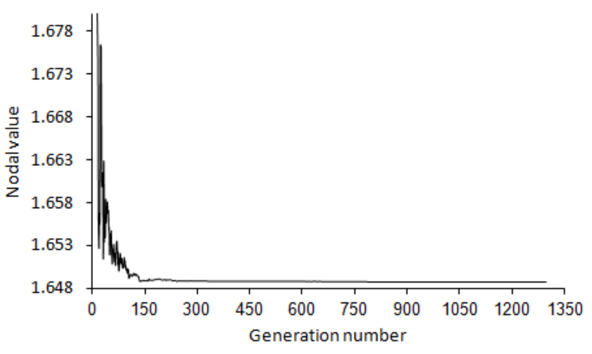
Dependent variable	Problem 1	Problem 2	Problem 3
$u_1(x)$	28%	31%	29%
$u_2(x)$	29%	30%	32%

Table 3 The influence of the step size on the convergence speed and the corresponding error of $u_1(x)$ for Problem 2.

Step size	Average generations	Maximum absolute error	Maximum absolute residual
0.1	1359	$7.68619311 \times 10^{-8}$	$5.41550512 \times 10^{-7}$
0.05	1671	$9.97232193 \times 10^{-9}$	$1.11413003 \times 10^{-8}$
0.025	1986	$6.79438253 \times 10^{-10}$	$7.30971949 \times 10^{-9}$
0.0125	2346	$7.18387063 \times 10^{-11}$	$4.09574688 \times 10^{-10}$

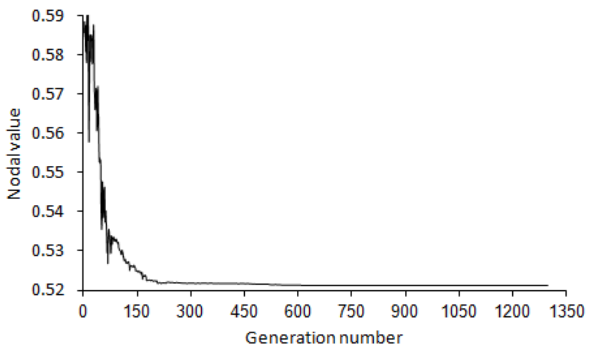


(a)

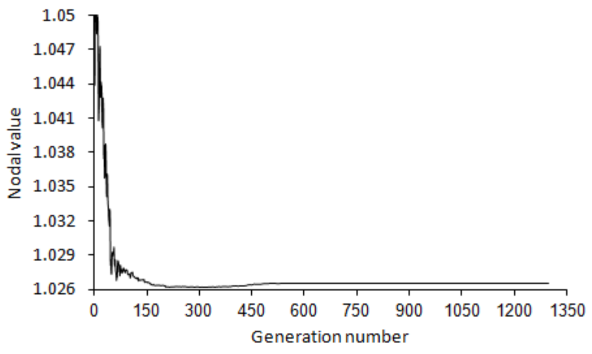


(b)

Fig. 5 Evolution of the nodal values of $u_1(x)$ for Problem 3 across all generations at: (a) the first nodal; (b) the fifth nodal.



(a)



(b)

Fig. 6 Evolution of the nodal values of $u_2(x)$ for Problem 3 across all generations at: (a) the fifth nodal; (b) the ninth nodal.

Table 4 The influence of the step size on the convergence speed and the corresponding error of $u_2(x)$ for Problem 2.

Step size	Average generations	Maximum absolute error	Maximum absolute residual
0.1	1359	$6.52892144 \times 10^{-8}$	$4.47232193 \times 10^{-7}$
0.05	1671	$9.18337204 \times 10^{-9}$	$1.00726841 \times 10^{-8}$
0.025	1986	$5.05074811 \times 10^{-10}$	$5.25578188 \times 10^{-9}$
0.0125	2346	$4.90500904 \times 10^{-11}$	$2.15255136 \times 10^{-10}$

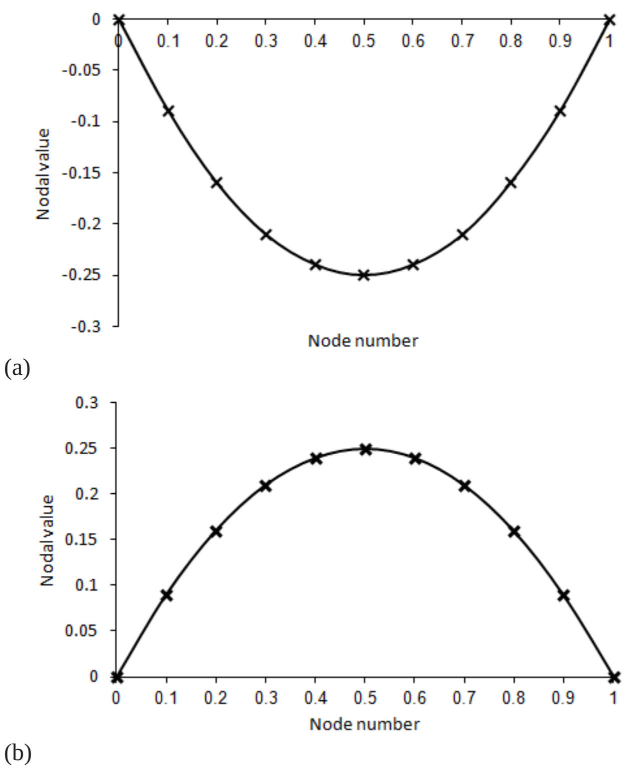


Fig. 7 Plot of the exact and approximate solutions for Problem 1, ——— exact solution and × × × approximate solution: (a) $u_1(x)$; (b) $u_2(x)$.

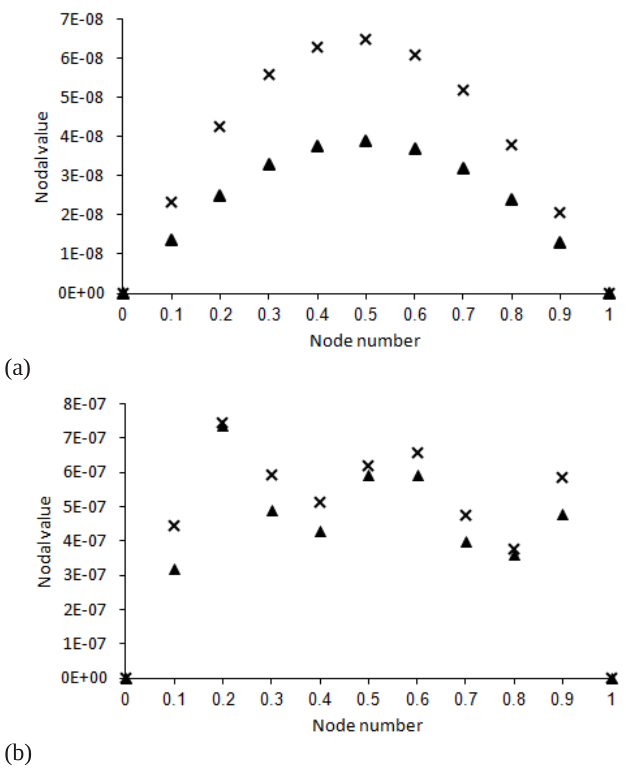


Fig. 8 Plot of the absolute error and absolute residual for Problem 1: (a) ▲▲▲ absolute error of $u_1(x)$ and × × × absolute error of $u_2(x)$; (b) ▲▲▲ absolute residual of $u_1(x)$ and × × × absolute residual of $u_2(x)$.

Numerical comparisons for Problem 2 are studied next. The conventional numerical methods that are used for comparison with CGA include the following: sinc-collocation method [4], reproducing kernel method [6], combination of homotopy perturbation and reproducing kernel methods [7], cubic B-spline scaling functions method [9]. Tables 5 and 6, show a comparison between the absolute errors of our method together with other aforementioned methods with 20 points for the first three methods and 33 points for the latest method. As it is evident from the comparison results, it was found that our method in comparison with the mentioned methods is much better with a view to accuracy and utilization.

The detailed data of $u_1(x)$ and $u_2(x)$ for Problem 3 that includes the exact nodal values, the CGA nodal values, the absolute error, and the absolute nodal residuals is given in Tables 7 and 8, respectively. It is clear that the accuracy obtained using CGA is moderate since it has a truncation error of the order $O\left(h^{10}\right)$.

The influence of the population size on the convergence speed of CGA is studied next for Problem 3 as shown in Table 9. The population size is increased in steps of 100 starting with 100 and ending with 1000. Small population sizes suffer from larger number of generations required for convergence and the probability of being trapped in local minima, while large population size suffer from larger number of fitness evaluations that means larger execution time. However, it is noted that the improvement in the convergence speed becomes almost

Table 5 Numerical comparison of $u_1(x)$ for problem 2 using CGA with other methods.

Node	Method of [4]	Method of [6]	Method of [7]	Method of [9]	Present method
0.08	1.4×10^{-4}	5.0×10^{-4}	7.7×10^{-5}	5.4×10^{-10}	3.1×10^{-9}
0.24	4.4×10^{-5}	1.4×10^{-3}	2.2×10^{-4}	1.2×10^{-9}	8.0×10^{-9}
0.40	6.7×10^{-5}	2.1×10^{-3}	3.3×10^{-4}	2.2×10^{-9}	8.7×10^{-9}
0.56	9.3×10^{-5}	2.2×10^{-3}	3.7×10^{-4}	2.4×10^{-9}	9.5×10^{-9}
0.72	4.9×10^{-5}	1.8×10^{-3}	3.1×10^{-4}	5.8×10^{-10}	7.3×10^{-9}
0.88	8.6×10^{-5}	9.0×10^{-4}	1.5×10^{-4}	3.4×10^{-10}	3.4×10^{-9}
0.96	7.1×10^{-5}	3.0×10^{-4}	5.4×10^{-5}	1.6×10^{-10}	1.2×10^{-9}

Table 6 Numerical comparison of $u_2(x)$ for problem 2 using CGA with other methods.

Node	Method of [4]	Method of [6]	Method of [7]	Method of [9]	Present method
0.08	2.4×10^{-4}	2.0×10^{-3}	7.1×10^{-4}	1.3×10^{-8}	1.8×10^{-9}
0.24	2.3×10^{-3}	5.6×10^{-3}	1.9×10^{-3}	9.9×10^{-9}	1.9×10^{-9}
0.40	8.9×10^{-4}	7.9×10^{-3}	2.7×10^{-3}	3.5×10^{-8}	2.5×10^{-9}
0.56	1.4×10^{-3}	8.2×10^{-3}	2.8×10^{-3}	1.2×10^{-7}	5.0×10^{-9}
0.72	3.1×10^{-3}	6.5×10^{-3}	2.2×10^{-3}	1.0×10^{-7}	1.8×10^{-9}
0.88	1.6×10^{-3}	3.1×10^{-3}	1.7×10^{-3}	4.9×10^{-8}	1.5×10^{-9}
0.96	9.8×10^{-4}	1.0×10^{-3}	3.6×10^{-4}	5.8×10^{-9}	5.9×10^{-10}

Table 7 Numerical results of $u_1(x)$ for Problem 3.

Node	Exact value	Approximate value	Absolute error	Absolute residual
0	1	1	0	0
0.1	1.10517092	1.10517092	$8.54004085 \times 10^{-10}$	$4.98986805 \times 10^{-8}$
0.2	1.22140276	1.22140276	$8.42495202 \times 10^{-10}$	$6.02052782 \times 10^{-8}$
0.3	1.34985881	1.34985881	$7.01441771 \times 10^{-10}$	$2.85466838 \times 10^{-8}$
0.4	1.49182470	1.49182470	$6.47288927 \times 10^{-10}$	$1.81271125 \times 10^{-8}$
0.5	1.64872127	1.64872127	$5.74708121 \times 10^{-10}$	$3.89186569 \times 10^{-8}$
0.6	1.82211880	1.82211880	$4.00564908 \times 10^{-10}$	$5.78639569 \times 10^{-8}$
0.7	2.01375271	2.01375271	$2.54064751 \times 10^{-10}$	$5.56192320 \times 10^{-8}$
0.8	2.22554093	2.22554093	$1.75550117 \times 10^{-10}$	$5.10268438 \times 10^{-8}$
0.9	2.45960311	2.45960311	$1.42755674 \times 10^{-11}$	$3.42025281 \times 10^{-8}$
1	2.71828183	2.71828183	0	0

negligible (saturation is reached) after a population size of 500.

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 termination condition of the algorithm and its value is set between 0.1 and 0.0000000001. Table 10 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 combined effect of the curve crossover probability, p_{cc}, and the curve mutation probability, p_{mc}, on the convergence speed of the algorithm and on the average fitness for Problem 3 are shown in Figure 9. The probability value is increased in steps of 0.2 starting with 0.1 and ending with 0.9 for both p_{cc} and p_{mc}, where the individual crossover probability and individual mutation probability are kept at 0.9. It is clear from the figures that when the probabilities values p_{cc} and p_{mc} are increasing gradually, the average number of generation required for convergence is increasing as well, while the average fitness is decreasing. Indeed, it can be seen that the curve

Table 8 Numerical results of $u_2(x)$ for Problem 3.

Node	Exact value	Approximate value	Absolute error	Absolute residual
0	0	0	0	0
0.1	0.10016675	0.10016676	$5.04412133 \times 10^{-9}$	$1.19781737 \times 10^{-7}$
0.2	0.20133600	0.20133601	$7.08781948 \times 10^{-9}$	$1.74816625 \times 10^{-7}$
0.3	0.30452029	0.30452030	$6.98424446 \times 10^{-9}$	$1.01331349 \times 10^{-7}$
0.4	0.41075233	0.41075233	$6.14133705 \times 10^{-9}$	$7.45835232 \times 10^{-8}$
0.5	0.52109531	0.52109531	$5.11226279 \times 10^{-9}$	$1.06886851 \times 10^{-7}$
0.6	0.63665358	0.63665359	$3.91191735 \times 10^{-9}$	$1.11482510 \times 10^{-7}$
0.7	0.75858370	0.75858370	$2.69370010 \times 10^{-9}$	$1.01986328 \times 10^{-7}$
0.8	0.88810598	0.88810598	$1.57728076 \times 10^{-9}$	$1.21174600 \times 10^{-7}$
0.9	1.02651673	1.02651673	$4.99999639 \times 10^{-10}$	$8.89829783 \times 10^{-8}$
1	1.17520119	1.17520119	0	0

Table 9 The effect of the population size on the convergence speed of CGA for Problem 3.

N_p	Average generations	Average fitness	Average error	Average residual
100	1654	0.99941551	$1.09565517 \times 10^{-6}$	$3.26623322 \times 10^{-5}$
200	1598	0.99971946	$4.75971286 \times 10^{-8}$	$1.56296010 \times 10^{-6}$
300	1496	0.99986403	$2.65161347 \times 10^{-8}$	$7.56412489 \times 10^{-7}$
400	1384	0.99998299	$3.32386734 \times 10^{-9}$	$9.45034882 \times 10^{-8}$
500	1299	0.99999339	$2.41761536 \times 10^{-9}$	$7.75241930 \times 10^{-8}$
600	1279	0.99999447	$1.29091708 \times 10^{-9}$	$3.62849959 \times 10^{-8}$
700	1272	0.99999466	$1.07896381 \times 10^{-9}$	$3.52085425 \times 10^{-8}$
800	1270	0.99999502	$8.67201518 \times 10^{-10}$	$2.43221070 \times 10^{-8}$
900	1268	0.99999554	$4.47235053 \times 10^{-10}$	$9.14527547 \times 10^{-9}$
1000	1264	0.99999587	$3.89764215 \times 10^{-10}$	$8.01199580 \times 10^{-9}$

Table 10 The influence of the maximum nodal residual on the convergence speed and the corresponding error for Problem 3.

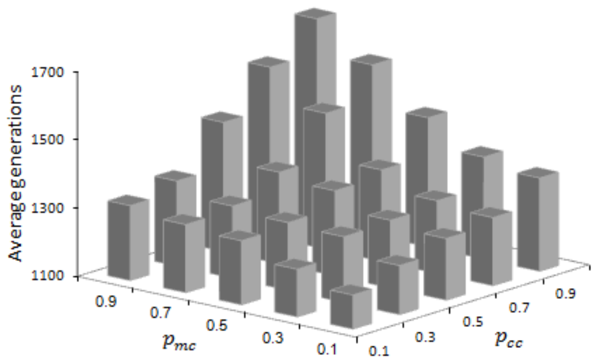
Maximum nodal residual	Average generations	Average fitness	Average error	Average residual
0.1	150	0.66715254	$2.45134428 \times 10^{-4}$	$2.77170881 \times 10^{-3}$
0.01	246	0.91864235	$1.69499232 \times 10^{-6}$	$4.92016253 \times 10^{-5}$
0.001	350	0.99857452	$1.57791166 \times 10^{-7}$	$8.03385433 \times 10^{-6}$
0.0001	728	0.99988219	$1.93835094 \times 10^{-8}$	$6.55284220 \times 10^{-7}$
0.00001	1204	0.99998013	$4.10215107 \times 10^{-9}$	$1.10423467 \times 10^{-7}$
0.000001	1256	0.99998032	$3.64324483 \times 10^{-9}$	$1.09332016 \times 10^{-7}$
0.0000001	1271	0.99998654	$2.92550326 \times 10^{-9}$	$9.47701704 \times 10^{-8}$
0.00000001	1294	0.99999049	$2.68850679 \times 10^{-9}$	$8.28563642 \times 10^{-8}$
0.000000001	1305	0.99999494	$6.98376995 \times 10^{-10}$	$7.47558559 \times 10^{-8}$
0.0000000001	1322	0.99999604	$4.28455537 \times 10^{-10}$	$2.20159822 \times 10^{-8}$

crossover probability and the curve mutation probability have a minor effect on the performance of the CGA.

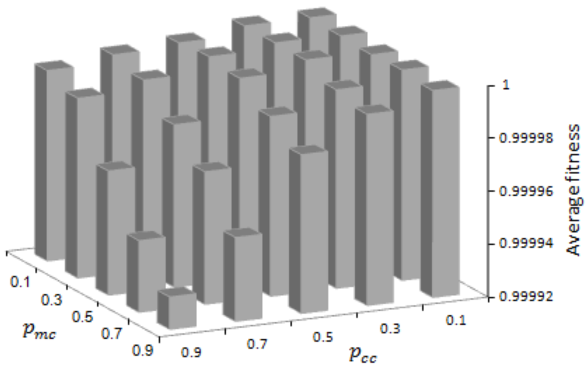
Finally, the effect of the different types of initialization methods on the convergence speed of the algorithm is discussed next. Three initialization methods are investigated in this work; the first method uses the MNGF, the second uses the MTHF, while the third is the mixed-type initialization method that initializes the first half of the population using the MNGF and the second half of the population using the MTHF. Table 11 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 and 2, the MNGF results in the fastest convergence speed while for Problem 3, the mixed-type initialization method results in the fastest convergence speed. In general, 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 and 2 is close to the MNGF. 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

Table 11 Convergence speed of CGA using different initialization functions.

Initialization method	Problem 1	Problem 2	Problem 3
MNGF	1241	1206	1452
MTHF	1653	1526	1395
Mixed-type	1503	1359	1299



(a)



(b)

Fig. 9 Plot of the combined effect of p_{cc} and p_{mc} for Problem 3 on: (a) convergence speed; (b) average fitness.

result, the mixed-type initialization method is used as the algorithm default method [16,17,18,19,20].

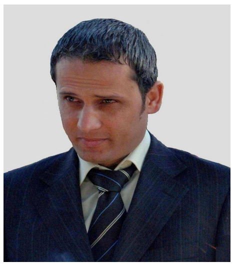
6 Concluding remarks

The main concern of this work has been to propose an efficient algorithm for the solutions of nonlinear system of second-order BVPs (1) and (2). The main goal has been achieved by introducing the CGA to solve this class of differential equations. We can conclude that the CGA is powerful and efficient technique in finding approximate solution for linear and nonlinear systems of second-order BVPs. In the proposed algorithm, each of the derivatives is replaced by an appropriate difference quotient approximation, where two smooth solution curves are used for representing the required nodal values. There is an important point to make here, the results obtained by the CGA are very effective and convenient in linear and nonlinear cases with less computational generation and less time. On the other aspects, the influence of different parameters, including the initialization method, the evolution of nodal values, the maximum nodal residual, the population size, the curve's probabilities, and the step size is also studied.

References

[1] S. Bellew and E. O'Riordan, Applied Numerical Mathematics, **51**, 171(2004).
[2] S. Matthews, E. O'Riordan and G. I. Shishkin, Journal of Computational and Applied Mathematics, **145**, 151 (2002).
[3] C. Xenophontos and L. Oberbroeckling, Applied Mathematics and Computation, **187**, 1351 (2007).
[4] M. Dehghan and A. Saadatmandi, Mathematical and Computer Modeling, **46**, 1434 (2007).
[5] H. B. Thompson and C. Tisdell, Applied Mathematics Letters, **15**, 761 (2002).
[6] F. Z. Geng and M. G. Cui, Journal of Mathematical Analysis and Applications, **327**, 1167 (2007).
[7] F. Z. Geng and M. G. Cui, Journal of Computational and Applied Mathematics, **235**, 2405 (2011).
[8] A. Saadatmandia, M. Dehghan and A. Eftekharia, Nonlinear Analysis: Real World Applications, **10**, 1912 (2009).
[9] M. Dehghan and M. Lakestani, International Journal of Computer Mathematics, **85**, 1455 (2008).
[10] J. F. Lu, Computers and Mathematics with Applications, **54**, 1133 (2007).
[11] A. S. Bataineh, M. S. M. Noorani and I. Hashim, Communications in Nonlinear Science and Numerical Simulation, **14**, 430 (2009).
[12] N. Caglar and H. Caglar, Computers and Mathematics with Applications, **57**, 757 (2009).
[13] M. Al-Smadi, O. Abu Arqub and N. Shawagfeh, Applied Mathematical Sciences, **6**, 2453 (2012).
[14] O. Abu Arqub and A. El-Ajou, Journal of King Saud University (Science), **25**, 73 (2013).
[15] A. El-Ajou, O. Abu Arqub and Shaher Momani, Discrete Dynamics in Nature and Society, **2012**, 1 (2012).
[16] Z. 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, Pakistan), (2002).
[17] Z. Abo-Hammour, N. Mirza, S. Mirza and M. Arif, Robotics and Autonomous Systems, **41**, 179 (2002).

[18] Z. Abo-Hammour, M. Yusuf, N. Mirza, S. Mirza, M. Arif and J. Khurshid, Int. J. Numer. Meth. Engng., **61**, 1219 (2004).
[19] Z. Abo-Hammour, A. Al-Asasfeh, A. Al-Smadi and O. Alsmadi, Optim. Control Appl. Meth., **32**, 414 (2010).
[20] O. Abu Arqub, Numerical Solution of Fuzzy Differential Equation using Continuous Genetic Algorithms (PhD. Thesis, University of Jordan, Jordan), (2008).
[21] J. Li, Journal of Computational and Applied Mathematics, **183**, 29 (2005).
[22] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning (Addison-Wesley: Reading, MA), (1989).
[23] D. Sakar and J. M. Modak, Bioprocess and Biosystems Engineering, **26**, 295 (2004).
[24] Y. C. Sim, S. B. Leng and V. Subramaniam, International Journal of Systems Science, **31**, 83 (2000).
[25] Y. Davidor, Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization (World Scientific: Singapore), (1991).
[26] P. Hajela and C. Y. Lin, In: Computational Engineering Using Metaphors from Nature, B. H. V. Topping (Eds.), (Edinburgh, UK), **77**, (2000) .
[27] K. D. M. Harris, R. L. Johnston and B. M. Kariuki, Acta Cryst., **A54**, 632 (1998).
[28] P. Cong P and T. Li, Analytica Chimica Acta, **293**, 191 (1994).



Omar Abu-Arqub received his Ph.D. from the university of Jordan (Jordan) in 2008. He then began work at Al-balqa applied university in 2008 as assistant professor of applied mathematics until now. His research interests focus on numerical analysis, optimization techniques, fractional calculus theory, and fuzzy differential and integral equations.

Zaer Abo-Hammour received his Ph.D. degree from the Quiad-Azam university (Pakistan) in 2002. He is an associate professor at the Mechatronics engineering department, the university of Jordan. His research interests are focused on control systems, robotics, genetic algorithms, and numerical fields. He is also a specialist in solar energy systems and fuel efficiency improvements.

Shaher Momani received his Ph.D from the university of Wales (UK) in 1991. He then began work at Mutah university in 1991 as assistant professor of applied mathematics and promoted to full Professor in 2006. He left Mutah university to the university of Jordan in 2009 until now. His research interests focus on the numerical solution of fractional differential equations in fluid mechanics, non-Newtonian fluid mechanics and numerical analysis. Prof. Momani has written over 200 research papers and warded several national and international prizes. Also, he was classified as one of the top ten scientists in the world in fractional differential equations according to ISI web of knowledge.