© 2017 IJSRSET | Volume 3 | Issue 8 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Solution of Higher Order Linear Boundary Value Problem Using Stochastic Method

Kavindra Soni, Dr. Mahendra Singh Khidiya
Department of Mechanical Engineering, College of Technology and Engineering, MPUAT, Udaipur, India

ABSTRACT

The aim of this work is to find the solution of higher order linear boundary value problem using genetic algorithm. A continuous genetic algorithm has been design and applied to the solution of higher order boundary value problem. The genetic algorithm solves the differential equation by a process of evaluating the best fittest solutions curve from a family of randomly generated solution curves. This method is applicable to differential equation of any order. Numerical results presented in the work illustrate the applicability of the genetic algorithm for any order linear boundary value problem.

Keywords : Genetic algorithm, stochastic method, higher order differential equation, centre-difference formula

I. INTRODUCTION

Most fundamental laws of science are based on models that explain variation in physical properties and state of system described by differential equations. To find the solution of differential equation is difficult job. The difficulty is increase as the order of differential equation and nonlinearity is increase. Therefore, solving the higher order differential equation is perhaps the most difficult problem in all of numerical computation.

The work presented in this paper is motivated by the success of using continuous genetic algorithm for solution of second-order, two-point linear and nonlinear boundary value problems. Although there are many possible methods or techniques are available for solving linear and nonlinear differential equation problems, such as, finite element method, soothing method, Ritz energy technique, etc. but they are difficult to implement or they required some more advanced mathematical tools. That tools are may be root finding technique or solution of some initial value problem.

Although many of the current methods for solving ordinary differential equations were developed around the turn of the century, the past 15 years or so has been a period of intensive research. The emphasis of this survey is on the methods and techniques for solving ordinary differential equations are based on genetic algorithms. The application of genetic algorithm in the field of numerical analysis is not new, for solving fluid flow problem genetic algorithm used by Pryor [1] and Pryor and Cline [2]. D. A. Diver [3] introduces a genetic algorithm for the solution of linear and nonlinear ordinary differential equation. Chaudhury and Bhattacharyya [4] used genetic algorithm to solve the Schrödinger equation. A hybrid scheme of genetic algorithm and Newton's method for solving a system of nonlinear equation introduced by Kare et al [5]. Raudensky et al [6] present a genetic algorithm for solving the one-dimensional inverse heat conduction problem. A novel method based on continuous genetic algorithm is introduced by Z. S. Abo-hammour et al [7] for the solution of the second-order, two-point boundary value problem. Advanced continuous genetic algorithm and their application in the motion planning of robotic manipulators are proposed by Z. S. Abo-Hammour et al [8]. A numeric genetic algorithm is introduced by Cong P. Li T. [9]. M. W. Gutowski proposed a smooth genetic algorithm[10] applied for finding the distribution of magnetic nanocrystallites.

II. Continuous Genetic Algorithm

2.1 Continuous genetic algorithm: The solution curves of boundary value problems are smooth in nature. So, continuous genetic algorithm is used. The construction of a genetic algorithm is based on the following conditions-

1. The genetic representation of potential problem solution,
2. A method for creating initial population of solution,
3. The design of the genetic operators,
4. The definition of the fitness function,
5. The setting of system parameters

Each of the above components greatly affects the solution obtained as well as the performance of the genetic algorithm. The following genetic operators that are used in this work.

2.2 Initialization: The implementation of a genetic algorithm starts with generating a population of possible solutions. For the solution of boundary value problem the initialization function is smooth and it should satisfied the given boundary values. Two smooth initialization function: the Gaussian function and the tangent hyperbolic function are used [7].

2.2.1 Tangent hyperbolic function: The tangent hyperbolic function is used in this work is given below. This function has some limitation for a particular type of boundary condition. If the nodal values of the extreme end are same, it will not work. In that case, the Gaussian function is applied, which is described in next section.

$$P(i,j) = 0.5\left(1 + \tanh\left(\frac{i-\mu}{\sigma}\right)\right)$$

For all $1 \le i \le N$ and $1 \le j \le N_p$

Where *P(i,j)* is the value of the i^{th} node for the j^{th} parent. μ is a random number within the range [11/4, 33/4] and it specified the centre of the function, σ is a random number within the range [1, 11/3] and it specifies the degree of dispersion [7]. *Np* is the number of initial population. The convergence speed is depending on the initialization function. If the initialization is closer to the final solution the convergence speed is faster and from study it is seen that after few generations convergence speed is governed by the selection, crossover and mutation operators.

2.2.2 Gaussian function: The Gaussian function which is used in this work for special boundary condition, is given as [7].

$$P(i,j) = \exp\left(\frac{-(i-\mu)^2}{2\sigma^2}\right)$$

For all $1 \le i \le N$ and $1 \le j \le N_p$

Where *P(i,j)* is the value of the i^{th} node for the j^{th} parent, μ is a random number within the range [11/4, 33/4] and it specified the centre of the function, σ is a random number within the range [1, 11/3].

2.3 Evaluation: Evaluation is performed by the mean of fitness function. It is a measure of quality of the individual in the population. The fitness function is defined as, $F = \frac{1}{1+R}$

Where *R* is overall residual. The main goal of the genetic algorithm is maximize the fitness function *F*.

2.4 Selection: The population is arranged in ascending order based on their relative fitness function value. The selection of population is rank based. The maximum fitness function values have highest rank and minimum fitness function values have lowest value. The bottom 50% of the population is discarded and the remaining 50% are selected for reproduction. The overall quality of the population is depending on the selection mechanism and its increase from one generation to the next.

2.5 Crossover: Crossover is the genetic algorithm operator that attempts to mix each pair of population selected as parents, to create the likelihood of keeping the good properties of each parent population in the offspring children [7]. Crossover provides the means by which valuable information is shared among the population. Crossover operator is implemented in several works in the literature. In this work crossover operator is expressed as

$$C_L(i) = w(i)P(i,j) + \big(1 - w(i)\big)P(i,k) \qquad C_{L+1}(i) = (1 - w(i))P(i,j) + w(i)P(i,k)$$

$$w(i) = 0.5\left(1 + \tanh\left(\frac{i-\mu}{\sigma}\right)\right)$$

For all $1 \le i \le N$

Where *P(i,j), P(i,k), C_L and C_{L+1}* represent the two parents chosen from the mating pool and the two

children obtained through crossover operator respectively, *w* represent the crossover weighting function within the range [1, 11/3]. The crossover operator also maintains the smoothness of the solution curves.

2.6 Mutation: Mutation operator has two important roles during the evaluation process in a genetic algorithm. First is to introduce unexplored genetic material to the population. Second is to maintain the diversity of the candidate solution in a population over the generations, preventing premature convergence of the genetic algorithm to suboptimal solution [7]. In this work mutation operator is expressed as

$$C^m(i,j) = C(i,j) + dM(i) \qquad M(i) = \exp\left(\frac{-(i-\mu)^2}{2\sigma^2}\right)$$

For all $1 \leq i \leq N$, $1 \leq j \leq N_p$

Where *C(i,j)* represent the j^{th} child produced through the crossover process, $C^m(i,j)$ is the mutated j^{th} child, *M* is the Gaussian mutation function and *d* represent a random number within the range [-1,1].

2.7 Elitism operator: After evaluation process, the population is going to change in every generation because of the crossover and mutation operators. Due to this the best information or population is vanish. To overcome that problem elitism operator is applied. Elitism operator insures that in the next generation the best-fitted individual is not less than previous fitted individual.

2.8 Extinction and immigration operator: After few numbers of generations, the population is going to stagnate, due to repetitions of the crossover and mutation operator. To overcome this problem extinction and immigration operator is applied to the population. In this process, half of the population is generated by same initial population function [7]. The another half population is generated by this formula that is given as

$$P(i,j) = P(i,1) + dM(i)$$

For $N_p/2+1 \leq j \leq N_p$

Where *P(i,j)* is the j^{th} parent generated by above operator, *P(i,1)* represent the best fitted population, *M* and *d* has already described in previous section.

2.9 Scaling operator: For solving higher order linear boundary value problem by genetic algorithm, after few numbers of generations the shape of the curve is generated but the exact solution is far away. This operator is given as $P^s(i,j) = P(i,j) \times B$

For all $1 \leq i \leq N$ and $1 \leq j \leq N_p$

Where $P^s(i,j)$ is the j^{th} parent generated by scaling operator, $P(i,j)$ represent the previous generated population and *B* is a random number between F_{max} and 1. F_{max} is the maximum value of fitness function so far found during the evaluation process.

2.10 Replacement: After the application of genetic operators to the initial (parent) population, a new population is generated. The previous population is replaced by new generated population, which is better fitted. This is also known as "life cycle" of the population.

2.11 Termination: The process described is iterated until an acceptable solution is found. The termination criterion is defined by the user, it could be either the difference in fitness value of few subsequent generations or a fixed number of generations which the user thinks, would provide a reasonable acceptable solution. In this work, the maximum fitness value is set to be 0.99999 and the maximum number of generation is set to be 500000. The genetic algorithm is terminating when one of the above criterion is met.

III. Problem Formulation and Numerical Results

Genetic algorithm and its operators are described in previous section are coded in MATLAB (R 2011a) for the solution of boundary value problem. In problem formulation, there are two types of parameter, one is genetic algorithm related and another is boundary value related. These parameters are described in next section. After that, the numerical results are shown in graphical and tabular form.

3.1 Genetic algorithm related parameter : In this work, following parameters are used in each problem. The initial population size (N_p) is set to be 100. The selection mechanism is rank based. The crossover and mutation probability is set to be 0.5 and 0.4 respectively. The number of elite parents, which are directly goes to next generation without any applications of genetic

operators, is set to be 10. The scaling, extinction and immigration operator is alternatively applied after every 100 number of generations.

The genetic algorithm is stopped when one of the following conditions is satisfied. These are as following

1. When the value of maximum fitness (F_{max}) is reach to 0.99999.
2. When the number of generation is, exceed by 500000.

3.2 Boundary value related parameter: Genetic algorithm does not require information of derivatives, because it is an optimization tool. Due to this, the governing differential equation is converting into discretization form. The centred-difference formulas, with truncation error of order *0* (h^2) is used to convert differential equation into discretization form.

The general fourth-order differential equation two-point boundary value problem of the form

$$y'''' = f(x, y, y', y'', y''') \qquad A \le x \le B$$

Together with boundary conditions

$y(A)=a \;\; y'(A)=b \qquad y''(B)=c \;\; y'''(B)=d$
$y(A)=a \;\; y''(A)=b \qquad y(B)=c \; y''(B)=d$

For the approximation, each derivative term is replaced in the discretized form by a difference quotient. The interval of the boundary value problem is equally partitioned into N+1 subintervals. The length of each subinterval is given as

$$h = \frac{B - A}{N + 1}$$

Where N = 9 is a number of interior nodes.
For approximating $y'(x_i)$, $y''(x_i)$, $y'''(x_i)$ *and* $y''''(x_i)$, the centre-difference formula is given as

$$y''''(x_i) \approx \frac{y_{i+2} - 4y_{i+1} + 6y_i - 4y_{i-1} + y_{i-2}}{h^4}$$

$$y'''(x_i) \approx \frac{y_{i+2} - 2y_{i+1} - 2y_{i-1} + y_{i-2}}{2h^3}$$

$$y''(x_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

$$y'(x_i) \approx \frac{y_{i+1} - y_{i-1}}{2h}$$

With the help of above equation the original differential equation is rewrite in discretized form as follows:

$$y_i = g(y_{i+2}, y_{i+1}, y_{i-1}, y_{i-2}, x_i)$$

The residual of i^{th} node (r_i) and the overall individual residual (R) is given as

$$r_i = y_i - g(y_{i+2}, y_{i+1}, y_{i-1}, y_{i-2}, x_i)$$

$$R = \sqrt{\sum_{i=1}^{N} r_i^2}$$

To convert the minimization problem of R into a maximization problem of F a mapping of the individual residual R is required. Therefore, the fitness function is defined as

$$F = \frac{1}{1 + R}$$

The maximization of above fitness function is the main task of genetic algorithm. The boundary value related parameters are as follows:

1. The number of interior nodes (N = 9).
2. The step size (h = 0.1).
3. The boundary conditions at both the ends that are vary from problem to problem.

The interval between which the differential equation is solved ($0 \le x \le 1$).

IV. Numerical Results

4.1 Case-1

Euler-Bernoulli beam equation is solving to find the deflection of a beam. In this case, one end of the beam is fixed and another end is free as shown in figure 3.1.
$\frac{d^4y}{dx^4} = \frac{f(x)}{EI}$ $\quad 0 \le x \le 1$ with following boundary condition

$y(0) = 0 \quad y'(0) = 0 \quad y''(0) = 0 \quad y'''(0) = 0$

Where: E = Modulus of Elasticity (material property), I = Moment of Inertia (geometry of material), $f(x)$ = load per unit length, $y(x)$ = deflection (displacement) from vertical and L = length of the beam

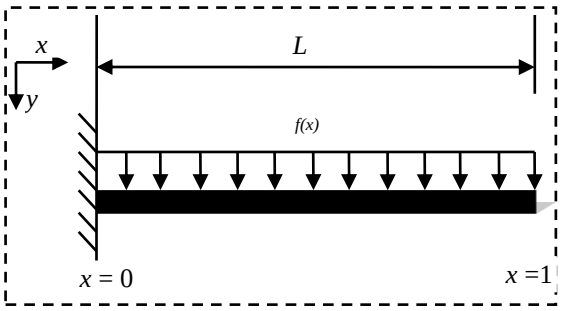

Fig. 4.1 Cantilever beam with uniformly distributed load

The discretization form of above governing differential equation is given as

$$y_i = \frac{\left(\frac{f(x)h^4}{EI} - y_{i+2} + 4y_{i+1} + 4y_{i-1} - y_{i-2}\right)}{6}$$

To find the exact information of any point we need information of four other points in centre-difference method. In this case, the value of node number 1 is known and all other all unknown.

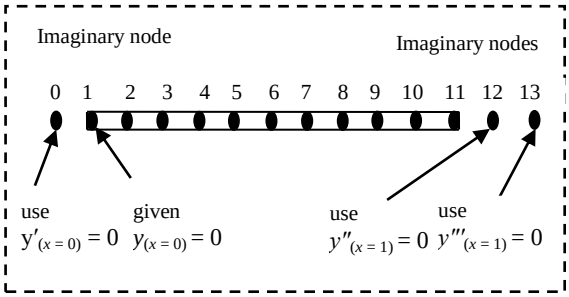

Fig. 4.2 Cantilever beam with discretization

For node 2 to 11 information is generated initially by tangent hyperbolic function and for the information of nodes 0, 12 and 13 is founded by the help of given boundary conditions. For getting the information of any node by centre-difference formula, the information of four other nodes is needed.

For example, for getting the information of node number 2, the information of node 0, 1, 3 and 4 is needed. But the genetic algorithm is only generating the information of 2 to 11 number of nodes. To overcome this problem the given boundary conditions are also convert into discretization form. The boundary condition in discretization form is give additional information for the solution curve. The genetic algorithm is coded in MATLAB (R 2011a) and the numerical results are compared with exact solution.

These results are shown for *f(x)/EI* = 0.1, 1 and 10 in figure 4.3, 4.4 and 4.5 respectively.

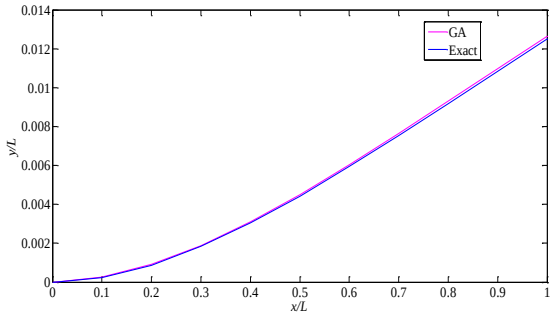

Fig. 4.3 Graph between exact value and GA value for *f(x)/EI* = 0.1

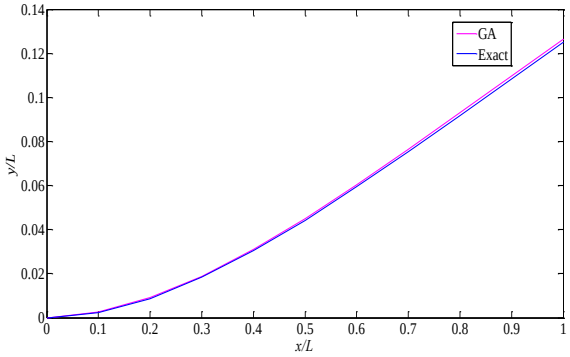

Fig. 4.4 Graph between exact value and GA value for *f(x)/EI* = 1

Table 4.1 Comparison between exact and GA value for case-1.

f(x)/E	**0.1**		**1**	
Node	**Exact**	**GA**	**Exact**	**GA**
1	0	0	0	0
2	0.0002	0.0002	0.0023	0.0025
3	0.0008	0.0009	0.0087	0.0090
4	0.0018	0.0018	0.0183	0.0188
5	0.0030	0.0031	0.0304	0.0310
6	0.0044	0.0045	0.0442	0.0451
7	0.0059	0.0060	0.0594	0.0603
8	0.0075	0.0076	0.0753	0.0764
9	0.0091	0.0093	0.0917	0.0930
10	0.0108	0.0109	0.1083	0.1097
11	0.0125	0.0126	0.1250	0.1265
Fitnes	0.99999		0.99999	

In given loading value the genetic algorithm terminated before it reached the maximum number of generation (500000). The fitness function reached the maximum fitness value of 0.99999. The number of generation after which the genetic algorithm terminated is given as in table 4.2

4.2 Case-2

In this case Euler-Bernoulli beam with both ends are simply support is solved by genetic algorithm. The governing differential equation is same as in case-1. The boundary condition at the extreme end is different from the case-1. For this case, the Gaussian function is use for initial population generation and all other genetic operators are to be same as used in case-1.

$$\frac{d^4y}{dx^4} = \frac{f(x)}{EI} \qquad 0 \le x \le 1$$

With following boundary condition

$y(0)=0 \;\; y''(0)=0 \;\; y(1)=0 \;\; y''(1)=0$

In this case, the exact information of two extreme end is known (i.e. deflection at node 1 and 11 is known). The information of nodes 0 and 12 is getting from boundary conditions and centre difference formula. After that, the genetic algorithm is applied for the solution of above linear fourth-order differential equation. The GA results shown in graphical and tabular form compare with exact solution.

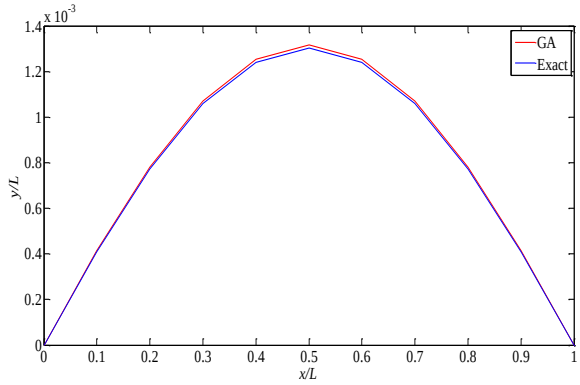

Fig. 4.11 Graph between exact value and GA value for $f(x)/EI = 0.1$

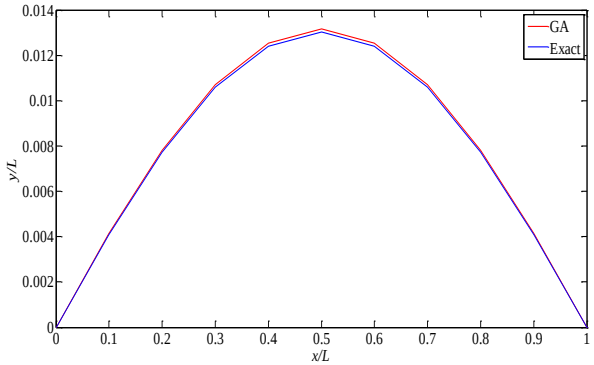

Fig. 4.12 Graph between exact value and GA value for $f(x)/EI = 1$

Table 4.2 Number of generation when GA is terminated for case-1

f(x)/EI	**Number of generation**
0.1	292605
1	287601

Table 4.3 Comparison between exact and GA value for case-2.

f(x)/EI	**0.1**		**1**	
Node (*i*)	**Exact**	**GA**	**Exact**	**GA**
1	0	0	0	0
2	0.0004	0.0004	0.0040	0.0041
3	0.0007	0.0007	0.0077	0.0078
4	0.0010	0.0010	0.0106	0.0106
5	0.0012	0.0012	0.0124	0.0125
6	0.0013	0.0013	0.0130	0.0131
7	0.0012	0.0012	0.0124	0.0125
8	0.0010	0.0010	0.0106	0.0106
9	0.0007	0.0007	0.0077	0.0078
10	0.0004	0.0004	0.0040	0.0041
11	0	0	0	0
Fitness	0.99999		0.99999	

In given loading value the genetic algorithm terminated before it reached the maximum number of generation (500000). The genetic algorithm solution is well match with exact solution. The fitness function reached the

maximum fitness value of 0.99999. The number of generation after which the genetic algorithm terminated is given as in table 4.4

Table 4.4 Number of generation when GA is terminated for case-2

f(x)/EI	**Number of generation**
0.1	270200
1	270000

V. CONCLUSION

The fourth-order linear boundary value problems are successfully solved by genetic algorithm. Genetic algorithm uses the objective function information and not the derivative information. Diversity is essential to the genetic algorithm because it enables the algorithm to search a large region of the population. For the solution of boundary value problem the solution curve must be smooth in nature. The genetic algorithm is versatile in nature because the operators of genetic algorithm are user and problem dependent and easy to excess. The numerical results from genetic algorithm are closely matched with available results.

VI. REFERENCES

[1]. Pryor RJ, Using a genetic algorithm to solve fluid flow problems. Transactions of the American Nuclear Society 1990; 61:435.

[2]. Pryor RJ, Cline DD. Use of a genetic algorithm to solve two-phase fluid flow problems on an Ncube multiprocessor computer. SAND92-2847C, Sandia National Laboratories, 1992.

[3]. Diver DA, Applications of genetic algorithms to the solution of ordinary differential equations. Journal of Physics A Mathematical and General 1993; 26(14):3503-3513.

[4]. Chaudhury P, Bhattacharyya SP, Numerical solutions of the Schrodinger equation directly or perturbatively by a genetic algorithm: test cases. Chemical Physics Letters 1998; 296(1-2):51-60.

[5]. Karr CL, Weck B, Freeman LM, Solutions to systems of nonlinear equations via a genetic algorithm. Engineering Applications of Artificial Intelligence 1998; 11(3):369-375.

[6]. Raudensky M, Woodbury KA, Kral J, Brezina T. Genetic algorithm in solution of inverse heat-conduction problems. Numerical Heat Transfer Part B-Fundamentals 1995; 28(3):293-306.

[7]. Abo-Hammour ZS, Yusuf M, Mirza MN, Mirza SM, Arif M and Khurshid J, Numerical solution of second-order, two-point boundary value problem using continuous genetic algorithm, International Journal for Numerical Methods in Engineering 2004; 61:1219-1242.

[8]. Abo-Hammour ZS. Advanced continuous genetic algorithms and their applications in the motion planning of robotic manipulators and the numerical solution of boundary value problems. University of Jordan, Ph.D. Thesis, 2002.

[9]. Cong P, Li T. Numeric genetic algorithm: part I. Theory, algorithm and simulated experiments. Analytica Chimica Acta 1994; 293:191-203.

[10]. Gutowski Morm W. Smooth genetic algorithm. Journal of Physics A Mathematical and General 1994; 27: 7893-7904.

[11]. Abo-Hammour ZS, Mirza NM, Mirza SM, Arif M, Cartesian path planning of robot manipulators using continuous genetic algorithms, Robotics and Autonomous Systems 2002; 41(4):179-223.

[12]. Goldberg DE. Genetic Agorithms in Search, Optimization, and Machine Learning. Addison-Wesley: New York, 1989.