A Metaheuristic Optimization Algorithm for Solving Higher-Order Boundary Value Problems

Shaza Alturky, Aleppo University, Syria*

https://orcid.org/0000-0003-3711-0644

George Albert Toma, Aleppo University, Syria

ABSTRACT

An effective metaheuristic algorithm to solve the higher-order boundary value problems, called a genetic programming technique, is presented. In this paper, a genetic programming algorithm, which depends on the syntax tree representation, is employed to obtain the analytical solutions of higher-order differential equations with the boundary conditions. The proposed algorithm can produce an exact or approximate solution when the classical methods lead to unsatisfactory results. To illustrate the efficiency and accuracy of the designed algorithm, several examples are tested. Finally, the obtained results are compared with the existing methods such as the homotopy analysis method, the B-Spline collocation method, and the differential transform method.

KEYWORDS

Boundary Conditions, Boundary Value Problems, Fitness Evaluation, Genetic Programming, Higher-Order Differential Equations, Metaheuristic, Population-Based Algorithm, Tree-Based Representation

INTRODUCTION

In recent years, higher-order boundary value problems (BVPs) have received much attention from researchers and scientists due to the fact that most phenomena in engineering, physiology and astrophysics are modeled by higher-order BVPs. Although there are various analytical and numerical methods for solving higher-order boundary value problems, these classical methods have their own disadvantages such as the initial approximation, low accuracy and a large number of required integrals. Therefore, many researchers are interested in exploring alternative methods to those traditional techniques; some of which are metaheuristic algorithms.

The genetic programming (GP), which is one of the metaheuristic algorithms, belongs to the population-based techniques and simulates the Darwin’s principle in evolution and natural selection.

Many research literatures dealing with higher-order boundary value problems, including:

The power series approximation method (Njoseh & Mamadu, 2016) that has been proposed for the numerical solution of a generalized linear and non-linear higher order BVPs. The homotopy analysis method (Siddiqi and Ifikhar, 2013a) is used for solving seventh-, eighth-, and tenth-order BVPs to obtain the approximate solutions. Siddiqi and Ifikhar (2013b) used the variation of parameter method
to find the analytical solutions of the seventh-order BVPs. The non-polynomial spline technique to solve the eighth-order boundary value problem is presented by Siddiqi and Twizell (2007).

Inc & Evans (2004) introduced the solutions of the eighth-order boundary value problems depending on the Adomian decomposition method.


In this research work, a tree-based genetic programming is utilized for higher-order boundary value problems where the BVP is formulated as an unconstrained optimization problem.

To design the genetic programming for solving this optimization problem, firstly, represent each candidate solution to the problem as a syntax-tree. A fitness function is constructed as sum square errors for the differential equation with its boundary conditions, where the fitness function is necessary for evaluating and comparing the quality of different trial solutions.

Moreover, GP forms an initial population of randomly generated possible solutions, and then iterates and evolves this population using the crossover and mutation operators.

The advantages of the designed genetic programming are listed as follows:

§ The genetic programming does not require the numerical calculation of integrals as in the differential transform method.
§ The nonlinear differential equation is not converted into a linear differential equation as in some classical techniques.
§ In the genetic programming, the initial approximation is not required like the homotopy analysis method.
§ The effective exploration of the search space by a population of trial solutions and not by a single solution as in the numerical methods.
§ In the tree-based genetic programming, the candidate solutions are expressed as syntax trees with different shapes and sizes, which gives more diverse solutions compared to the genetic algorithm that is based on fixed length chromosomes.
§ The genetic programming leads to the analytical solution without approximating the solution as a polynomial or a power series as in the genetic algorithm.
Due to these characteristics, the tree-based genetic programming algorithm has attracted more attention from the other techniques when dealing with the higher-order differential equations.

The objectives of this paper are outlined as follows:

1. To take advantage of the tree-based genetic programming as an optimization tool for solving analytically higher-order BVPs.
2. To investigate the rapid convergence to the exact solution compared to the classical techniques.
3. To extend the application of the genetic programming as an optimization technique to find the best solution of the problem.
4. To suggest an efficient method as an alternative technique to classical methods.

This article is constructed as follows: Section 2 introduces a brief overview of metaheuristic algorithms. In Section 3, the basic concept of tree-based genetic programming is highlighted. Section 4 provides the formulation of the problem for the higher-order boundary value problems. Section 5 exposes several numerical examples to demonstrate the power of the presented algorithm. In addition, the comparisons with some existing methods are tabulated. Finally, a conclusion is covered in Section 6.

AN OVERVIEW OF METAHEURISTIC ALGORITHMS

In the last years, a new kind of approximation algorithms has emerged which tries to combine basic heuristic methods in higher-level frameworks aimed at efficiently and effectively exploring a search space. These methods are nowadays called metaheuristics. The term metaheuristic, first introduced in (Glover, 1986), derives from the composition of two Greek words. Heuristic derives from the verb heuriskein that means “discover or find”, while the suffix meta means “beyond or in an upper level”.

In the fields of computer science and mathematical optimizations, the term “metaheuristic” represents a higher-level procedure to find, search, generate, or select a heuristic that may provide a good solution to an optimization problem.

The fundamental properties that characterize metaheuristic are summarized as follows (Almufti, 2019):

- Metaheuristics are strategies that guide the search process.
- The goal is to efficiently explore the search space in order to find near–optimal solutions.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- Metaheuristics are not problem-specific.
- Today more advanced metaheuristics use search experience (embodied in some form of memory) to guide the search.

Metaheuristics are broadly classified into two categories; single solution-based and population-based algorithms. Single solution-based algorithms are those in which a solution is randomly generated and improved until the optimum result is obtained, whereas population-based algorithms are those in which a set of solutions are randomly generated in a given search space and solution values are updated during iterations until the best solution is generated (Almufti, 2019).

Figure 1 exposes the classification of metaheuristic algorithms.

The single solution-based algorithms may trap the local optima that may be a hinder to find the global optimum. In contrast, the population-based algorithms have an inherent ability to escape local optima (Almufti, 2019). Due to this, the population-based algorithms have attracted the attention of multitudinous academicians and researchers.
Genetic programming (GP) is introduced by Dr. John R. Koza in 1992. The GP is a systematic method for getting computers to automatically solve a problem starting from a high-level statement of what needs to be done. The GP iteratively transforms a population of computer programs into a new generation of programs by applying the genetic operations. The genetic operations include crossover, mutation, and reproduction.

Genetic programming is an extension of the genetic algorithm (Holland 1975) in which the structures in the population are not fixed-length character strings that encode candidate solutions to a problem, but programs that, when executed, are the candidate solutions to the problem (Poli et al., 2008). Programs are expressed in the genetic programming as syntax trees rather than as lines of code. For example, the simple expression \((3 \ y + \sin(10 \ x))\) is represented in Figure 2.

The tree comprises nodes and links. The nodes indicate the instructions to execute. The links indicate the arguments for each instruction. In the following, the internal nodes in a tree will be called functions, while the tree’s leaves will be called terminals (Poli et al., 2008).

Note how the variables and constants in the program \((x, y, 1 \text{ and } 3)\), which are called terminals in the GP, are the leaves of the tree, while the arithmetic operations (+, *, and sin) are the internal nodes (typically called functions). The sets of allowed functions and terminals together form the primitive set of a GP system.

To apply a GP, these are often termed the five major preparatory steps. The key choices are (Poli et al., 2008):

1. The set of terminals:
   § Constants that can be pre-specified, randomly generated as part of the tree creation process, or created by mutation.
   § Variables (e.g., \(x, y\)).

2. The set of functions,
   § Arithmetic functions: +, -, *, /.
   § Mathematical functions: sin, cos, Exp, log, sqrt, ^, …
   § Boolean functions: And, Or, Not…
   § Loop functions: For-Repeat…

3. The fitness function for measuring the fitness values of individuals in the population.

4. Certain parameters for controlling the run, population size, maximum depth tree and the probabilities of performing the crossover & mutation.
5. **The termination criterion** which may include a maximum number of generations to be run as well as a problem-specific success predicate.

Algorithmically, a simple GP comprises the following executional steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Randomly create an initial population of programs from the available primitives.</td>
</tr>
<tr>
<td>2.</td>
<td>Repeat</td>
</tr>
<tr>
<td>3.</td>
<td>Execute each program and ascertain its fitness.</td>
</tr>
<tr>
<td>4.</td>
<td>Select one or two program(s) from the population with a probability based on fitness to participate in genetic operations.</td>
</tr>
<tr>
<td>5.</td>
<td>Create new individual program(s) by applying genetic operations with specified probabilities.</td>
</tr>
<tr>
<td>6.</td>
<td>Until stopping condition is met (e.g., reaching a maximum number of generations).</td>
</tr>
</tbody>
</table>

Figure 3 shows the flowchart of the genetic programming.

**Methodology**

In this section, the mechanism, of how the tree-based genetic programming works to solve higher-order boundary value problems, is provided step-by-step.

Consider the following boundary value problem:
The five major preparatory steps should be specified such that:

1. **Terminal set**: \( \{ x, \text{random constants} \} \)
2. **Function set**: \( \{ +, -, *, \div, \sin, \cos, ^, \sqrt{}, \log, \exp, \tan \} \)
3. **Fitness function (or Error Function)** (Viswanadham & Raju, 2012): The steps for the calculation of the fitness for any individual are:
   a) Choose \( N \) uniformly distributed points in the \([a, b]\)
   b) Calculate the error of differential equation:
   \[
   \varepsilon_1 = \varepsilon_1(2)
   \]
   c) Calculate the error of boundary conditions:
   \[
   \varepsilon_2 = \frac{1}{n} \sum_{k=1}^{n} \left( g_k^2 \left( x, M^{(i)}, \ldots, M^{(n-1)} \right) \right)^2 \bigg| x = a \text{ or } x = b. \tag{3}
   \]
   d) Calculate the fitness function:

\[
\varepsilon = \varepsilon_1 + \varepsilon_2
\]

Where \( N \) is the number of discretization of the interval \([0, 1]\), is the mean square errors of the differential equation (1), is the mean square errors of boundary conditions of (1), is the fitness function which represents the mean square errors of (1), subject to the availability of the parameters, such that in case of both, then the approximate results closer to the exact solution (Pandey & Verma, 2008).
According to above steps, the higher-order BVP is converted into an unconstrained optimization problem, and the objective is to obtain the best solution that makes the value of the fitness function minimized.

4. **Control Parameters**: Population size, Crossover probability, Mutation probability, Max depth tree.

5. **Stopping Condition**: maximum number of generations.

**NUMERICAL EXAMPLES**

To show the effectiveness of the presented tree-based genetic programming for solving five higher-order boundary value problems, several numerical examples are reported in this section. The produced solutions in this work are obtained relying on the windows form application (Visual C#). In addition, the obtained results are compared with the traditional methods.

5.1. Example (1)

Consider the following seventh-order boundary value problem (Siddiqi & Iftikhar, 2013b):

\[
\begin{align*}
    u^{(5)}(x) &= -u(x) - e^{-x}(35 + 12 \ x + 2 \ x^2), 0 \leq x \leq 1 \\
    u(0) &= 0, \ u(1) = 0, \\
    u^{(i)}(0) &= 1, \ u^{(i)}(1) = -e, \quad i = 1, 2, 3 \\
    u^{(3)}(0) &= 0, \ u^{(2)}(1) = -4e, \\
    u^{(3)}(1) &= -3
\end{align*}
\]  

(4)

Figure 4. Windows application C# for Example (1)
To solve this problem using the tree-based genetic programming, a windows form application Visual C# is designed as in Figure 4.

Note the above Figure 1, the optimal control parameters are provided and the output leads to the following best analytical solution:

\[ u(x) = xe^x - 1.50005 \cdot E - 30 \cdot x^{10} - x^2 e^x \]

Table 1 contains a comparison of the absolute errors between the presented GP with the homotopy analysis method (Siddiqi & Iftikhar, 2013a) and the variation of parameters method (Siddiqi & Iftikhar, 2013b).

### Table 2. Comparison of the absolute errors for Example (1)

<table>
<thead>
<tr>
<th>( x )</th>
<th>Designed GP</th>
<th>Homotopy analysis method (Siddiqi &amp; Iftikhar, 2013a)</th>
<th>Variation of parameters method (Siddiqi &amp; Iftikhar, 2013b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>1.500000500 E-40</td>
<td>5.39291 ( E - 14 )</td>
<td>8.55607 ( E - 13 )</td>
</tr>
<tr>
<td>0.2</td>
<td>2.000000000 E-38</td>
<td>4.85167 ( E - 14 )</td>
<td>9.94041 ( E - 12 )</td>
</tr>
<tr>
<td>0.3</td>
<td>8.857352952 E-36</td>
<td>3.92464 ( E - 14 )</td>
<td>3.52244 ( E - 11 )</td>
</tr>
<tr>
<td>0.4</td>
<td>1.572864524 E-34</td>
<td>2.21489 ( E - 14 )</td>
<td>7.3224 ( E - 10 )</td>
</tr>
<tr>
<td>0.5</td>
<td>1.000000000 E-33</td>
<td>3.84137 ( E - 14 )</td>
<td>1.08769 ( E - 10 )</td>
</tr>
<tr>
<td>0.6</td>
<td>9.069929423E-33</td>
<td>2.10831 ( E - 13 )</td>
<td>1.29035 ( E - 10 )</td>
</tr>
<tr>
<td>0.7</td>
<td>1.000000000 E-32</td>
<td>1.99785 ( E - 13 )</td>
<td>1.51466 ( E - 10 )</td>
</tr>
<tr>
<td>0.8</td>
<td>1.610613273 E-31</td>
<td>3.29736 ( E - 13 )</td>
<td>2.717974 ( E - 10 )</td>
</tr>
<tr>
<td>0.9</td>
<td>3.486801835 E-31</td>
<td>1.77622 ( E - 12 )</td>
<td>7.48179 ( E - 10 )</td>
</tr>
<tr>
<td>1.0</td>
<td>1.000005000 E-30</td>
<td>1.65159 ( E - 12 )</td>
<td>2.1729 ( E - 09 )</td>
</tr>
</tbody>
</table>

Giving the previous Table 1, the designed tree-based genetic programming has a high level of accuracy compared to the traditional analytical methods, where the precise performance of the proposed technique depended on absolute errors around 10^{-30} to 10^{-40}, which refers to the rapid convergence versus the homotopy analysis method (HAM) and the variation of parameters method.

### 5.2. Example (2)

Consider the following seventh-order nonlinear differential equation with given boundary conditions (Siddiqi & Iftikhar, 2013b):

\[ \text{Consider the following seventh-order nonlinear differential equation with given boundary conditions (Siddiqi & Iftikhar, 2013b):} \]
\[
\begin{aligned}
    u^{(2)}(x) &= u(x)u^{(1)}(x) + e^{-2x}(2 + e^x(x - 8) - 3x + x^2),
        0 \leq x \leq 1 \\
    u(0) &= 1, \\n    u(1) &= 0, \\
    u^{(1)}(0) &= -2, \\n    u^{(1)}(1) &= -\frac{1}{e}, \\
    u^{(2)}(0) &= 3, \\n    u^{(2)}(1) &= \frac{2}{e}, \\
    u^{(3)}(0) &= -4
\end{aligned}
\]  

(5)

To find the analytical solution of (5), a windows form application is executed as seen in the following Figure 5.

Figure 5. Windows application C# for Example (2)

The output of the above Figure 5 leads to the following optimal analytical solution:

\[u(x) = e^{-x} - xe^{-x} - 1.75 - 30x^{10}\]

Table 2 comprises a comparison of the absolute errors between the presented GP and the homotopy analysis method (Siddiqi & Iftikhar, 2013a).

The previous Table 2 shows that the designed genetic programming is more accurate than the homotopy analysis method.

5.3. Example (3)

The following seventh-order linear boundary value problem is considered (Siddiqi & Iftikhar, 2013):

\[
\begin{aligned}
    u^{(7)}(x) &= u(x)u^{(6)}(x) + e^{-2x}(2 + e^x(x - 8) - 3x + x^2),
        0 \leq x \leq 1 \\
    u(0) &= 1, \\n    u(1) &= 0, \\
    u^{(1)}(0) &= -2, \\n    u^{(1)}(1) &= -\frac{1}{e}, \\
    u^{(2)}(0) &= 3, \\n    u^{(2)}(1) &= \frac{2}{e}, \\
    u^{(3)}(0) &= -4
\end{aligned}
\]  

(5)
Table 3. Comparison of the absolute errors for Example (2)

<table>
<thead>
<tr>
<th>$x$</th>
<th>Designed GP</th>
<th>Homotopy analysis method (Siddiqi &amp; Iftikhar, 2013a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>1.750000000 E-40</td>
<td>4.15223 $E – 14$</td>
</tr>
<tr>
<td>0.2</td>
<td>1.792000000 E-37</td>
<td>4.18332 $E – 13$</td>
</tr>
<tr>
<td>0.3</td>
<td>1.033357500 E-35</td>
<td>1.21736 $E – 12$</td>
</tr>
<tr>
<td>0.4</td>
<td>1.835008000 E-34</td>
<td>1.95471 $E – 12$</td>
</tr>
<tr>
<td>0.5</td>
<td>1.000000000 E-33</td>
<td>2.03731 $E – 12$</td>
</tr>
<tr>
<td>0.6</td>
<td>1.058158080 E-32</td>
<td>1.37063 $E – 12$</td>
</tr>
<tr>
<td>0.7</td>
<td>4.943316858 E-32</td>
<td>4.66988 $E – 13$</td>
</tr>
<tr>
<td>0.8</td>
<td>1.879048192 E-31</td>
<td>4.8378 $E – 14$</td>
</tr>
<tr>
<td>0.9</td>
<td>6.101872702 E-31</td>
<td>6.00561 $E – 14$</td>
</tr>
<tr>
<td>1.0</td>
<td>1.750000000 E-30</td>
<td>1.29172 $E – 15$</td>
</tr>
</tbody>
</table>

Figure 6. Windows application C# for Example (3)
$$\begin{align*}
    u^{(2)}(x) &= xu(x) + e^x (x^2 - 2x - 6), 0 \leq x \leq 1 \\
    u(0) &= 1, u(1) = 0, \\
    u^{(i)}(0) &= 0, u^{(i)}(1) = -e, \\
    u^{(2)}(0) &= -1, u^{(2)}(1) = -2e, \\
    u^{(i)}(0) &= -2
\end{align*}$$

(6)

To implement the presented genetic programming GP, a windows form application in visual C# is designed as in Figure 6. In Figure 6, the best control parameters of tree-based genetic programming are achieved and used to solve the related higher-order BVP, which the output gives the following optimal analytical solution:

$$u(x) = -xe^x - 5.017e^{-30}x^{10} + e^x$$

Table 3 presents a comparison of the absolute errors between the presented GP with the homotopy analysis method (Siddiqi & Iftikhar, 2013a) and differential transform method (Siddiqi & Iftikhar, 2013b).

<table>
<thead>
<tr>
<th>$x$</th>
<th>Designed GP</th>
<th>Homotopy analysis method (Siddiqi &amp; Iftikhar, 2013a)</th>
<th>Differential Transform Method (Siddiqi &amp; Iftikhar, 2013b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>1.750000000 E-40</td>
<td>3.41727 E-13</td>
<td>4.6585 E-13</td>
</tr>
<tr>
<td>0.2</td>
<td>1.792000000 E-37</td>
<td>6.25056 E-14</td>
<td>5.7126 E-12</td>
</tr>
<tr>
<td>0.3</td>
<td>1.033357500 E-35</td>
<td>1.42442 E-13</td>
<td>2.1299 E-11</td>
</tr>
<tr>
<td>0.4</td>
<td>1.835008000 E-34</td>
<td>8.83738 E-14</td>
<td>4.6995 E-11</td>
</tr>
<tr>
<td>0.5</td>
<td>1.708984375 E-33</td>
<td>6.43929 E-14</td>
<td>7.4307 E-11</td>
</tr>
<tr>
<td>0.6</td>
<td>1.058158080 E-32</td>
<td>1.51812 E-12</td>
<td>8.9219 E-11</td>
</tr>
<tr>
<td>0.7</td>
<td>4.943316858 E-32</td>
<td>1.47904 E-12</td>
<td>7.9767 E-11</td>
</tr>
<tr>
<td>0.8</td>
<td>1.879048192 E-31</td>
<td>4.94338 E-12</td>
<td>4.6686 E-11</td>
</tr>
<tr>
<td>0.9</td>
<td>6.101872702 E-31</td>
<td>5.3817 E-12</td>
<td>1.0960 E-11</td>
</tr>
<tr>
<td>1.0</td>
<td>1.750000000 E-30</td>
<td>1.20811 E-11</td>
<td>6.9252 E-16</td>
</tr>
</tbody>
</table>

Moreover, the tree-based GP leads to satisfactory results with higher accuracy compared to the HAM and the differential transform method based on the comparison shown in Table 3.
5.4. Example (4)

Consider the following tenth-order boundary value problem (Siddiqi & Iftikhar, 2013a):

\[
\begin{align*}
    u^{(10)}(x) &= \frac{14175}{4} + (x + u(x) + 1)^{11}, \quad 0 < x < 1 \\
    u(0) &= 0, \quad u(1) = 0, \\
    u^{(1)}(0) &= -\frac{1}{2}, \quad u^{(1)}(1) = 1, \\
    u^{(2)}(0) &= \frac{1}{2}, \quad u^{(2)}(1) = 4, \\
    u^{(3)}(0) &= \frac{3}{4}, \quad u^{(3)}(1) = 12 \\
    u^{(4)}(0) &= \frac{3}{2}, \quad u^{(4)}(1) = 48 \\
\end{align*}
\]

(7)

Figure 7 exhibits the implemented genetic programming GP to find the optimal solution of (7)

In Figure 7, the optimal control parameters of the tree-based genetic programming are reached, where the analytical solution of (7) is described as follows:

\[
u(x) = \frac{2}{2-x} - x - 1.75 E - 30 x^{13} + 7.012 E - 30 x^{17}
\]
The following Table 4 presents a comparison of the absolute errors between the presented GP with the homotopy analysis method (Siddiqi & Iftikhar, 2013a) and quintic B-spline collocation method (Viswanadham & Raju, 2012).

**Table 5. Comparison of the absolute errors for Example (4)**

<table>
<thead>
<tr>
<th>( x )</th>
<th>Designed GP</th>
<th>Homotopy analysis method (Siddiqi &amp; Iftikhar, 2013a)</th>
<th>Quintic B-spline collocation method (Viswanadham &amp; Raju, 2012)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>1.749298800 E-43</td>
<td>3.95413 E – 11</td>
<td>1.322478E – 06</td>
</tr>
<tr>
<td>0.2</td>
<td>1.424409231 E-39</td>
<td>7.33317 E – 10</td>
<td>4.231930E – 06</td>
</tr>
<tr>
<td>0.3</td>
<td>2.699512168 E-37</td>
<td>7.33317 E – 09</td>
<td>1.676381E – 05</td>
</tr>
<tr>
<td>0.4</td>
<td>1.053939877 E-35</td>
<td>6.06524 E – 09</td>
<td>4.245341E – 05</td>
</tr>
<tr>
<td>0.5</td>
<td>1.601257324 E-34</td>
<td>7.74775 E – 09</td>
<td>6.663799E – 05</td>
</tr>
<tr>
<td>0.6</td>
<td>1.098724094 E-33</td>
<td>6.56402 E – 09</td>
<td>6.940961E – 05</td>
</tr>
<tr>
<td>0.7</td>
<td>6.4352518 E-34</td>
<td>3.48667 E – 09</td>
<td>4.750490E – 05</td>
</tr>
<tr>
<td>0.8</td>
<td>6.16893557 E-32</td>
<td>9.23198 E – 10</td>
<td>1.643598E – 05</td>
</tr>
<tr>
<td>0.9</td>
<td>7.245774611 E-31</td>
<td>5.33521 E – 11</td>
<td>2.607703E – 07</td>
</tr>
<tr>
<td>1.0</td>
<td>5.262000000 E-30</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 8. Windows application C# for Example (5)
According to the above Table 4, the obtained results show that the presented GP is more convergent to the exact solution compared to the HAM and the Quintic B-Spline collocation method.

5.5. Example (5):
Consider the following twelfth-order boundary value problem (Viswanadham & Raju, 2012):

\[
\begin{align*}
  u^{(12)} (x) &= 2e^x u^{(2)} (x) + u^{(3)} (x) , \quad 0 \leq x \leq 1 \\
  u^{(2k)} (0) &= 1, \quad k = 0,1,2,3,4,5 \\
  u^{(2k)} (1) &= \frac{1}{e^k} , \quad k = 0,1,2,3,4,5
\end{align*}
\]  

(8)

To implement the presented genetic programming GP, a windows form application in visual C# is designed as in Figure. 8.

The optimal values of parameters for the designed GP and the higher-order differential equation with its corresponding boundary conditions are demonstrated in the above Figure 8.

The output of this application provides the following exact solution: \( u(x) = e^{-x} \) where, the fitness value is corresponding to zero.

CONCLUSION

The tree-based genetic programming technique is successfully applied to solve the higher-order boundary value problems analytically. The genetic programming is designed for dealing with different boundary value problems of the higher-order differential equations and results show that the proposed algorithm is an efficient, accurate and easy to implement approach. In addition, the precise performance of the proposed technique depended on the absolute error around \(10^{-30}\) to \(10^{-43}\), which refers to the rapid convergence versus the existing methods.

As a result, the tree-based genetic programming can produce satisfactory results with a high level of accuracy.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.
REFERENCES


Shaza Alturky is a lecturer at the Department of Mathematics, Faculty of Science, University of Aleppo. She holds a PhD degree in Applied Mathematics & Computer Science from Aleppo University-Syria in 2021. She is the author of tens of articles in metaheuristic algorithms and Boundary-Initial value problems. She is specialized in the Genetic Algorithm, Genetic Programming, Evolutionary Algorithms, Boundary-Initial Value Problems and Singular Differential Equations arising in Astrophysics & Physiology Applications.

George Albert Toma is a lecturer at the Aleppo University, Department of Mathematics, Faculty of Science. He holds a PhD degree in Applied Mathematics & Computer Science from Aleppo University-Syria since 2019. His research focuses essentially on the semi-analytical techniques, integral transform methods, integro-differential equations, partial differential equations appearing in engineering and hydrodynamic models. Recently, Dr. George Albert Toma invented a new integral transform method called Kharrat-Toma Transform technique to solve various boundary-initial value problems.