Global Analysis and Discrete Mathematics
Volume 2, Issue 1, pp. 75–89
ISSN: 2476-5341

A Genetic Programming-based Scheme for Solving Fuzzy Differential Equations

Ali Akbar Bani* · **Maliheh Darbani**

Received: 10 October 2016 / Accepted: 14 March 2017

Abstract This paper deals with a new approach for solving fuzzy differential equations based on genetic programming. This method produces some trial solutions and seeks the best of them. If the solution cannot be expressed in a closed analytical form then our method produces an approximation with a controlled level of accuracy. Furthermore, the numerical results reveal the potential of the proposed approach.

Keywords Genetic programming · Fuzzy differential equations · Grammatical evolution

Mathematics Subject Classification (2010) 34A07.

1 Introduction

Studying fuzzy differential equations paves the way for mathematical modeling of real world problems in which there are uncertainty or ambiguity. Generally fuzzy sets theory is a powerful instrument for modeling uncertainties and processing ambiguous information dealing with mathematical models which are used in real world. Uncertainty is an adjective for information [21] and fuzzy differential equations is a way to remove problems deriving from it. The concept of fuzzy derivation was firstly offered by Chang and Zadeh [7] and followed

*Corresponding author
Ali Akbar Bani
Department of Mathematics, Gorgan Branch, Gomishan Center, Islamic Azad University, Gomishan, Iran.
E-mail: bani_ali@yahoo.com

Maliheh Darbani
Department of Mathematics, Gorgan Branch, Gomishan Center, Islamic Azad University, Gomishan, Iran.

by Dubois and Prade [8]. They introduced a form of elementary fuzzy calculus programmed based on the extension principle [22].
Other methods have been described by Puri and Ralescu [18] and Goetschel Voxman [10]. Using fuzzy differential equations is a direct way of putting uncertainty to the dynamic systems. The method used by Kalva is based on Hukuhara difference and formulizes the concept of differential equations in a fuzzy environment [12]. To find fuzzy initial value, there are a lot of methods most of which are based on Hukuhara derivation. Some cases of these methods which are based on Hukuhara derivation suffer from a weakness which is that it leads to the answers with increasing their base length. These weaknesses have been removed by interpreting FDE as a family of the items of differential and the concept of augmented derivation has been introduced ([4],[5]). According to the new definition, it may be possible that FDE answer has decreasing base length. Some researchers suggest some other methods including number methods to find FDE answers ([1],[6],[9],[11],[14]) such as Runge Kutta method [2], Adomian method [3].
The current research seeks to solve FDE by genetic programming. In fact, for every $r \in [0,1]$ we change a FDE which is:

$$\acute{x} = f(t,x) \qquad , \qquad x(0) = x_0$$

to an ODE system which is:

$$x_r'^{-} = f_r^{-}(t, x_r^{-}, x_r^{+})$$

$$x_r'^{+} = f_r^{+}(t, x_r^{-}, x_r^{+})$$

And then we search for its answers by genetic programming.

2 Preliminaries

Let X be a nonempty set. A fuzzy set u in X is characterized by its membership function $u : X \to [0,1]$. Then $u(x)$ is the degree of membership of a element x in the fuzzy set u for each $x \in X$.

Definition 2.1. Let a fuzzy subset of the real line $u : R \to [0,1]$. Then u is a fuzzy number if it satisfies the following properties :

i) u is a normal, i.e. $\exists x_0 \in R$ with $u(x_0) = 1$.
ii) u is fuzzy convex (i.e. $u(tx + (1-t)y) \geq min\{u(x), u(y)\}, \forall t \in [0,1], x, y \in R$).
iii) u is upper semi continuous on R (i.e. $\forall \varepsilon > 0 \exists \delta > 0$ such that $u(x) - u(x_0) < \varepsilon, |x - x_0 < \delta|$).
iv) u is compactly supported i.e. $cl\{x \in R; u(x) > 0\}$ is compact.
Let us denote by F_R the space of fuzzy numbers.

Definition 2.2. For $0 < r \leq 1$, we denote

$$[u]_r = \{x \in R; u(x) \geq r\}$$

$$[u]_0 = cl\{x \in R; u(x) > r\}.$$

Then u_r will be called $r-$cut of fuzzy number u while u_0 is called the support of it. It is clear that if $u \in F_R$ is a fuzzy number and u_r is its $r-$cut then: u_r is a closed interval $u_r = [u_r^-, u_r^+]$ for any $r \in [0, 1]$.

Definition 2.3. A fuzzy number u is completely determined by pair $u = (u^-, u^+)$ of functions $u^-(r), u^+(r) : [0, 1] \to R$ satisfying the following conditions :

i) $u^-(r) = u_r^- \in R$ is a bounded, monotonic, increasing (nondecreasing) left-continuous function in $(0, 1]$ and it is right-continuous at 0.
ii) $u^+(r) = u_r^+ \in R$ is bounded, monotonic, decreasing (nonincreasing) left-continuous function in $(0, 1]$ and it is right-continuous at 0.
iii) For all $r \in (0, 1]$ we have : $u^-(r) \leq u^+(r)$.

For every $u, v \in F_R$ and $\lambda \in R, r \in [0, 1]$ defined addition and scalar multiplication as follows :

$$(u + v)_r = u_r + v_r$$

$$(\lambda.u)_r = \lambda u_r.$$

Definition 2.4. The distance between two arbitrary fuzzy numbers u, v to help their $r-$cuts is defined as follows :

$$d(u, v) = Sup \;\; max\{|u_r^- - v_r^-|, |u_r^+ - v_r^+|\}$$

with this definition and addition and scalar multiplication defined above (F_R, d) is a complete metric space.

Definition 2.5. Let $u, v \in F_R$. If there exists $w \in F_R$ such that $u = v + w$ then w is called the Hukuhara difference of u, v and it is denoted by $u \ominus v$.

Definition 2.6. Let function $f : R \longrightarrow F_R$ is a fuzzy number valued function. This function is called continuous if for every $\varepsilon > 0$ there exists a $\delta > 0$ such that :

$$|x - x_0| < \delta \Longrightarrow d(f(x), f(x_0)) < \varepsilon.$$

Definition 2.7. A function $f : (a, b) \longrightarrow F_R$ is called Hukuhara differentiable if for $h > 0$ sufficiently small the H-differences $f(x+h) \ominus f(x)$ and $f(x) \ominus f(x-h)$ exist and if there exist an element $f'(x) \in F_R$ such that

$$\lim \frac{f(x+h) \ominus f(x)}{h} = \lim \frac{f(x) \ominus f(x-h)}{h} = f'(x)$$

Then $f'(x)$ is called the fuzzy derivative of f at x.

3 Fuzzy differential equations

In this section, a first order fuzzy differential equation is described. Then it is replaced by its equivalent parametric form, and the new system which contains two ordinary differential equations, is solved.
A fuzzy differential equation with an initial condition is in the following form:

$$x'(t) = f(t, x(t)) \quad t \in [t_0, t_1] \tag{1}$$

It is clear that the fuzzy function $f(t,x)$ is the mapping $f : R \times F_R \longrightarrow F_R$. For the existence of a unique solution to (1), the following conditions are sufficient.
a) f is continuous
b) A Lipschitz condition $d(f(t,x), f(t,y)) \leq Ld(x,y)$ satisfied for some $L > 0, x, y \in F_R$.
Now we can to replace (1) by the following equivalent system :

$$x'^{-}(t) = f^{-}(t,x) = F(t, x^{-}, x^{+}) \quad x^{-}(t_0) = x_0^{-}$$

$$x'^{+}(t) = f^{+}(t,x) = G(t, x^{-}, x^{+}) \quad x^{+}(t_0) = x_0^{+}$$

where

$$F(t, x^{-}, x^{+}) = min\{f(t,v) | v \in [x^{-}, x^{+}]\}$$

$$G(t, x^{-}, x^{+}) = max\{f(t,v) | v \in [x^{-}, x^{+}]\}$$

For every $t \in [t_0, t_1]$ and $r \in [0,1]$, the parametric form is given by :

$$x'^{-}(t,r) = F(t, x^{-}(t,r), x^{+}(t,r)) \quad x^{-}(t_0, r) = x_0^{-}(r)$$

$$x'^{+}(t,r) = G(t, x^{-}(t,r), x^{+}(t,r)) \quad x^{+}(t_0, r) = x_0^{+}(r)$$

Now we choose a set of points $t_i, i = 0, 1, 2, \ldots, M-1$ in the interval $[t_0, t_1]$. Thus, we can write as:

$$x'^{-}(t_i, r) - F(t_i, x^{-}(t_i), x^{+}(t_i)) = 0 \qquad x^{-}(t_0, r) = x_0^{-}(r)$$

$$x'^{+}(t_i, r) - G(t_i, x^{-}(t_i, r), x^{+}(t_i, r)) = 0 \qquad x^{+}(t_0, r) = x_0^{+}(r).$$

4 Grammatical evolution

Grammatical evolution is an evolutionary automatic programming process that can generate programs in a desired language. The process needs the BNF grammar for production rule. It deals vectors of integers that called chromosomes. Each integer in chromosome indicates a production rule from BNF grammar. *GE* starts from the first symbol and the program string is then gradually generated by placement of non-terminal symbols with the right hand of the selected production rule. Algorithm has the following steps:

get an element from the chromosome (with value R)
select the rule according to the scheme

$$Rule = R\ mod\ SC.$$

Where SC is the number of rules for the specific non-terminal symbol.The process of replacing non-terminal symbols with the right hand of production rules is continued until either a full program has been generated or the end of chromosome has been reached. In the latter case we can reject the entire chromosome or we can start over (wrapping event) from the first element of the chromosome. In our approach we allow at most two wrapping events to occur.

In Fig.1 is given a small piece of the C programming language grammar. The numbers in parentheses denote the sequence number of the corresponding production rule to be used in the selection procedure described above.

The symbol S in the grammar denotes the start symbol of the grammar. For example, suppose we have the chromosome $x = [2, 6, 0, 4, 8, 7, 16, 1, 3, 1]$. In Table. 1 we show how a valid function is produced from t. The resulting function in the above example is $f(t) = 2t^2 + t$. Further details about grammatical evolution can be found in ([15],[16],[17],[19]).

Fig.1 The grammar of the proposed method

```
s ::=< expr > (0)

expr ::=< expr >< op >< expr >        (0)
          |(< expr >)                     (1)
          |func(< expr >)                 (2)
          | < digit >                     (3)
          |t                              (4)

< op >::= +                         (0)
          | −                         (1)
          | ∗                         (2)
          |/                          (3)

< func >::= t+                 (0)
          |t-                     (1)
          |t*                     (2)
          |t/                     (3)

< digit >::= 0         (0)
          |1              (1)
          |2              (2)
          |3              (3)
          |4              (4)
          |5              (5)
          |6              (6)
          |7              (7)
          |8              (8)
          |9              (9)
```

Table.1 Example of program construction

string	choromosome	opration
<expr>	2,6,0,4,8,7,16,1,3,1	2 mod 5=2
fun(<expr>)	6,0,4,8,7,16,1,3,1	6 mod 4=2
t*(<expr>)	0,4,8,7,16,1,3,1	0 mod 5=0
t*(<expr>,<op>,<expr>)	4,8,7,16,1,3,1	4 mod 5=4
t*(t,<op>,<expr>)	8,7,16,1,3,1	8 mod 4=2
t*(t+<expr>)	7,16,1,3,1	7 mod 5=2
t*(t+<func>(<expr>))	16,1,3,1	16 mod 4=0
t*(t+t+(<expr>))	1,3,1	1 mod 5=1
t*(t+t+(<expr>))	3,1	3 mod 5=3
t*(t+t+(<digit>))	1	1 mod 10=1
t*(t+t+1)	1	1 mod 10=1

5 Description of the algorithm

The proposed algorithm is established for polynomials. Solve a given fuzzy differential equation has the following phases:

1. Initialization.
2. Fitness evaluation.
3. Genetic operations.
4. Termination control.

5.1 **Initialization**

In this stage, some values are given to crossover rate and mutation rate. Based on the values, a fraction of chromosome are subject to changes specific to mutation and crossover and then go to the next generation.

5.2 **Fitness evaluation**

We know a fuzzy differential equation with an initial condition is in the following form:

$$x'(t) = f(t, x(t)) \qquad , \qquad x(t_0) = x_0 \qquad , \qquad t \in [t_0, t_1]. \tag{2}$$

We can to replace (2) by the following equivalent system :

$$x'^{-}(t) - F(t, x^{-}(t), x^{+}(t)) = 0 \qquad , \qquad x^{-}(t_0) = x_0^{-}$$

$$x'^{+}(t) - G(t, x^{-}(t), x^{+}(t)) = 0 \qquad , \qquad x^{+}(t_0) = x_0^{+}.$$

In other words,

$$f_1(t, x^{-}, x^{+}, x'^{-}) = 0 \qquad , \qquad x^{-}(t_0) = x_0^{-}$$

$$f_2(t, x^{-}, x^{+}, x'^{+}) = 0 \qquad , \qquad x^{+}(t_0) = x_0^{+}.$$

Now the proposed method can apply for above ordinary differential equations system. The steps for the fitness evaluation of the chromosome i are the following[20]:

a) Choose M equidistant points $(x_0, x_1, \ldots, x_{M-1})$ in according to the range.

b) For every chromosome i, split the chromosome uniformly in 2 parts. Any part is peculiar to a equation of system.

c) Construct the 2 models S_{i1}, S_{i2}.

d) Calculate the quantities

$$D(S_{ij}) = \sum (f_j(t_l, S_{i1}(t_l), S'_{i1}(t_l))^2 \qquad j = 1, 2.$$

e) Calculate the quantity $D(S_i) = D(S_{i1}) + D(S_{i2})$.

f) Calculate the associated penalties

$$P^-(S_{ij}) = \lambda(S_{ij}(t_0) - x_0^-) \qquad j = 1, 2.$$

$$P^+(S_{ij}) = \lambda(S_{ij}(t_0) - x_0^+) \qquad j = 1, 2.$$

where λ is a positive number.

g) Calculate the total penalty value

$$P(S_i) = P(S_{i1}) + P(S_{i2}).$$

h) Finally, the fitness of the chromosome i is given by:

$$Q_i = D(S_i) + P(S_i).$$

5.3 Genetic operations

The genetic operations used are crossover and the mutation. The crossover allow new individuals to be created. The individuals participating in the crossover operation are selected proportionate to fitness and via tournament selection i.e. the individual with the best fitness in the group is selected the others are discarded.

The crossover operation produces two offspring .The two offspring are usually different from their two parents and different from each other. In that operation for each couple of new chromosomes two parents are selected, we cut these parent-chromosomes at a randomly chosen point and we exchange the right-hand-side sub-chromosome [13].

The next genetic operator used is the mutation. Mutation is used very sparingly in work. The mutation operation in an asexual operation in that it operates on only one individual. It begins by randomly selecting a string from population and then randomly selecting a number between 1 and L (length of string) as the mutation point. Then the single character at the selected mutation point is changed.

5.4 Termination control

The genetic operators are used to produce new generations until either a chromosome is found in the population with the best fitness or the maximum number of frequency happens for producing new generation.

6 Examples

Example 6.1. Consider the following fuzzy initial value problem :

$$x'(t) = x(t) - t^2 + 1 \qquad x(0) = (-3\ ,\ -1\ ,\ 1)$$

We know the exact solution is $x(t) = (-4e^t + t^2 + 2t + 1\ ,\ -2e^t + t^2 + 2t + 1\ ,\ t^2 + 2t + 1)$. After solving with proposed method, we obtain the following results:

$$x(t) = (-3 - 2t - 0.747t^2 - \frac{2}{3}t^3 - 0.46t^4\ \ ,\ \ -1 - \frac{1}{3}t^3 - 0.103t^4\ \ ,\ \ t^2 + 2t + 1)$$

Each of exact and approximate solutions have been shown in Figure 2.

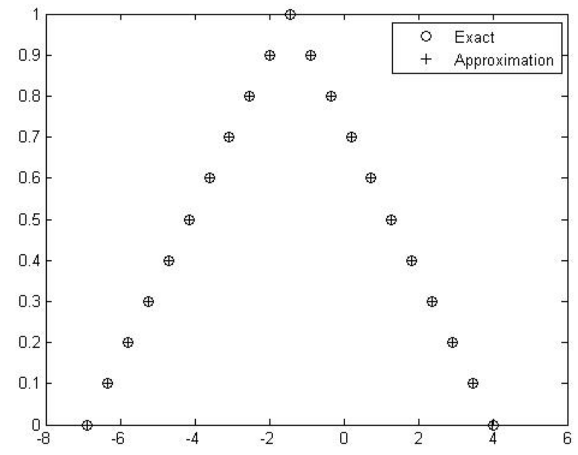

Figure 2.The exact and approximated solution for Example 6.1.

The numerical results can be seen in Table 2. Moreover, Figs. 3 shows the accuracy of the solution, $\underline{E}(1,r)$, $\overline{E}(1,r)$ (for $t=1$), respectively.

Table.2 Comparison of the exact $x_a(1,r)$ and approximated $x_T(1,r)$ solutions for Example 6.1.

r	$\underline{x}_a(1,r)$	$\underline{x}_T(1,r)$	$\underline{E}(t=1,r)$	$\overline{x}_a(1,r)$	$\overline{x}_T(1,r)$	$\overline{E}(t=1,r)$
0	-6.873127	-6.873710	5.828571e-4	4.000005	4.000010	5.436020e-6
0.1	-6.329475	-6.329969	4.941752e-4	3.456343	3.456378	3.499119e-5
0.2	-5.785822	-5.786228	4.054933e-4	2.912682	2.912747	6.454636e-5
0.3	-5.242170	-5.242486	3.168114e-4	2.369021	2.369115	9.410153e-5
0.4	-4.698517	-4.698745	2.281294e-4	1.825359	1.825483	1.236567e-4
0.5	-4.154864	-4.155004	1.394475e-4	1.281698	1.281851	1.532118e-4
0.6	-3.611212	-3.611263	5.076569e-5	0.738036	0.738219	1.827670e-4
0.7	-3.067559	-3.067521	3.791621e-4	0.194375	0.194587	2.123222e-4
0.8	-2.523907	-2.523780	1.265981e-4	0.349286	0.349044	2.418773e-4
0.9	-1.980254	-1.980039	2.152800e-4	0.892947	0.892676	2.714325e-4
1	-1.436601	-1.436297	3.039619e-4	1.436609	1.436308	3.009877e-4

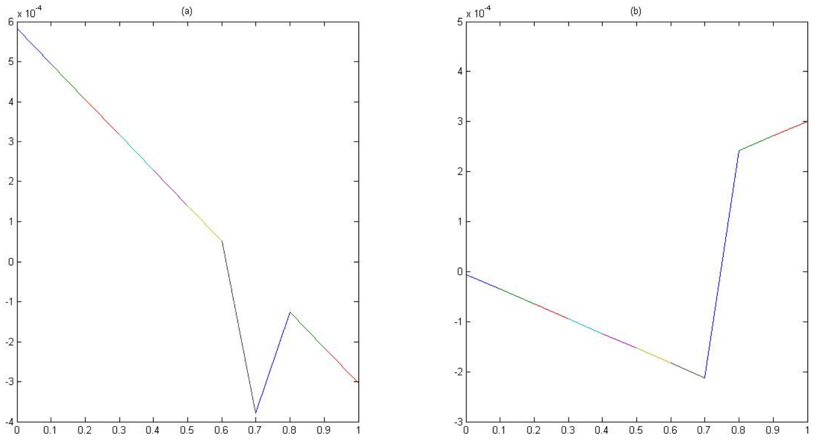

Figure 3. Figure (a) $\underline{E}(1,r)$ and figure (b) $\overline{E}(1,r)$ for Example 6.1.

Example 6.2. Consider the following fuzzy initial value problem:

$$x'(t) = -x(t) + t + 1 \qquad x(0) = (0.4\ ,\ 1\ ,\ 1.85)$$

We know the exact solution is $x(t) = (0.4e^{-t}+t\ ,\ e^{-t}+t\ ,\ 1.85e^{-t}+t)$. After solving with proposed method, we obtain the following results:

$$x(t) = (0.4{+}0.06t{+}3.8t^2{-}0.01t^3\ ,\ 1{+}0.15t{-}0.5t^2{-}0.15t^3\ ,\ 1.85{-}0.75t{+}0.25t^2{-}0.8t^4)$$

Each of exact and approximate solutions have been shown in Figure 4.

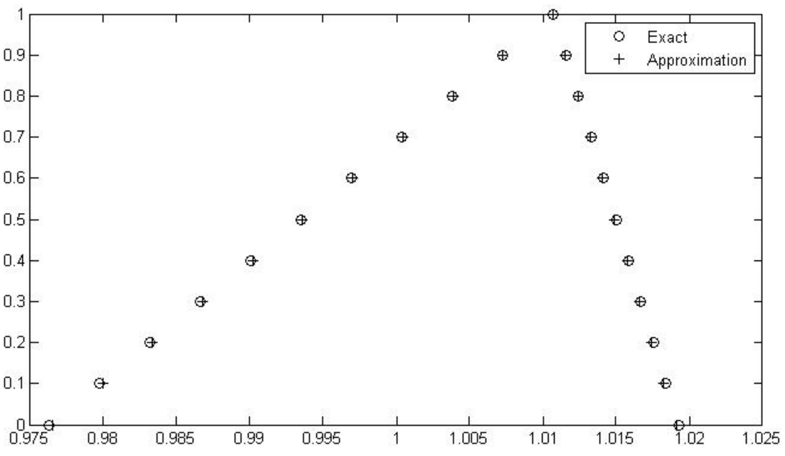

Figure 4. The exact and approximated solution for Example 6.2.

The numerical results can be seen in Table 3. Besides, Fig. 4 shows the accuracy of the solution, $\underline{E}(0.15, r)$, $\overline{E}(0.15, r)$ (for $t = 0.15$), respectively.

Table 3
Comparison of the exact $x_a(0.15, r)$ and approximated $x_T(0.15, r)$ solutions for Example 6.2.

r	$\underline{x}_a(0.15, r)$	$\underline{x}_T(0.15, r)$	$\lvert\underline{E}(t = 0.15, r)\rvert$	$\overline{x}_a(0.15, r)$	$\overline{x}_T(0.15, r)$	$\lvert\overline{E}(t = 0.15, r)\rvert$
0	0.494298	0.494499	2.009445e-4	1.742307	1.742694	3.874530e-4
0.1	0.545939	0.546119	1.809520e-4	1.669146	1.669494	3.478234e-4
0.2	0.597579	0.597740	1.610695e-4	1.595986	1.596294	3.081939e-4
0.3	0.649219	0.649360	1.409670e-4	1.522826	1.523094	2.685644e-4
0.4	0.700859	0.700980	1.209745e-4	1.449665	1.449894	2.289349e-4
0.5	0.752499	0.752600	1.009820e-4	1.376505	1.376694	1.893054e-4
0.6	0.804139	0.804220	8.098955e-5	1.303344	1.303494	1.496758e-4
0.7	0.855779	0.855840	6.099706e-5	1.230184	1.230294	1.100463e-4
0.8	0.907419	0.907460	4.100456e-5	1.157024	1.157094	7.041684e-5
0.9	0.959059	0.959080	2.101206e-5	1.083863	1.083894	3.078732e-5
1	1.010699	1.010700	1.019570e-6	1.010696	1.010694	1.526160e-6

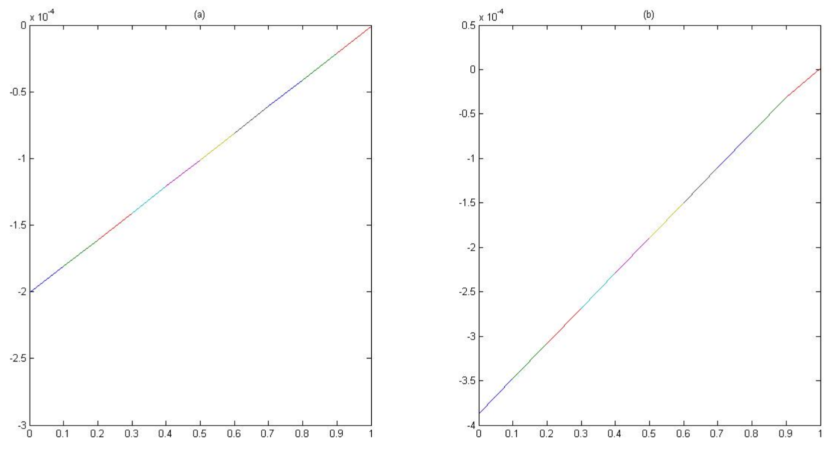

Figure 5. Figure (a) $\underline{E}(0.15, r)$ *and figure (b)* $\overline{E}(0.15, r)$ *for example 6.2.*

Example 6.3. Consider the following fuzzy initial value problem:

$$x'(t) = -x(t) + 2e^{-t} \qquad x(0) = (-1\ ,\ 0\ ,\ 1)$$

We know the exact solution is $x(t) = (e^{-t} - 2e^{t}\ ,\ 0\ ,\ 2e^{t} - e^{-t})$. After solving with proposed method, we obtain the following results:

$$x(t) = (-1 - 3t - 5.4t^3 - 3.2t^4 \quad , \quad 0 \quad , \quad 1 + 3t + 0.5t^2 + 0.6t^3)$$

Each of exact and approximate solutions have been shown in Figure 6.

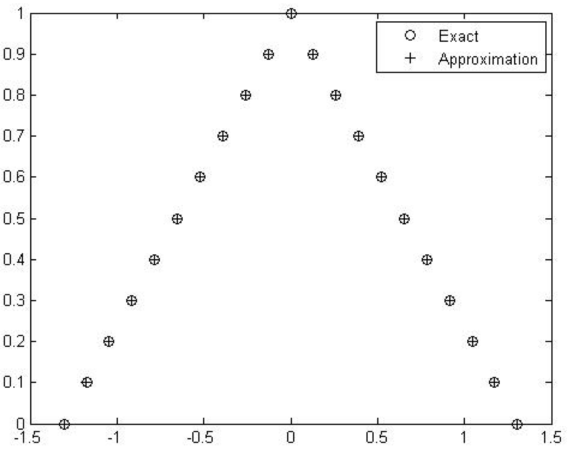

Figure 6.The exact and approximated solution for Example 6.3.

The numerical results can be seen in Table 4. Also, Figs. 7 show the accuracy of the solution, $\underline{E}(0.1, r)$, $\overline{E}(0.1, r)$ (for $t = 0.1$), respectively.

Table.4 Comparison of the exact $x_a(0.1, r)$ and approximated $x_T(0.1, r)$ solutions for Example 6.3.

r	$\underline{x}_a(0.1, r)$	$\underline{x}_T(0.1, r)$	$\lvert\underline{E}(t = 0.1, r)\rvert$	$\overline{x}_a(0.1, r)$	$\overline{x}_T(0.1, r)$	$\lvert\overline{E}(t = 0.1, r)\rvert$
0	-1.305500	-1.305704	2.045517e-4	1.305500	1.305602	1.022678e-4
0.1	-1.74950	-1.175134	1.840965e-4	1.174950	1.1750421	9.204105e-5
0.2	-1.044400	-1.044563	1.636413e-4	1.044400	1.044481	8.181427e-5
0.3	-0.913850	-0.913993	1.431861e-4	0.913850	0.913921	7.158748e-5
0.4	-0.783300	-0.783422	1.227310e-4	0.783300	0.783361	6.136070e-5
0.5	-0.652750	-0.652852	1.022758e-4	0.652750	0.652801	5.113391e-5
0.6	-0.522200	-0.522281	8.182068e-5	0.522200	0.522240	4.090713e-5
0.7	-0.391650	-0.391711	6.136551e-5	0.391650	0.391680	3.068035e-5
0.8	-0.261100	-0.261140	4.091034e-5	0.261100	0.261120	2.045356e-5
0.9	-0.130550	-0.130570	2.045517e-5	0.130550	0.130560	1.022678e-4
1	0	0	0	0	0	0

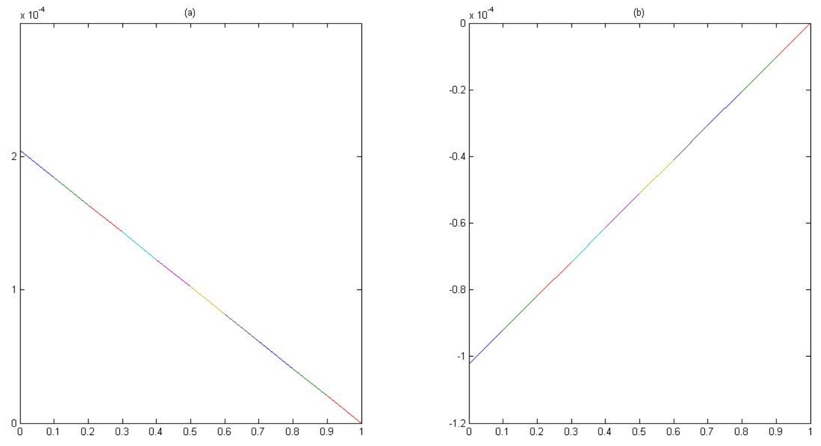

Figure 7. Figure (a) $\underline{E}(0.1, r)$ *and figure (b)* $\overline{E}(0.1, r)$ *for example 6.3.*

7 Experimental results

For the first algorithm, each chromosome is split uniformly in 2 parts. Each part of the chromosome represents the solution of the corresponding for control function and trajectory function. We applied 80% for the crossover rate and 5% for mutation rate. We investigated the matter of these two parameters by performance some experiments. Each experiment was run 25 times. The population size was set to 1000 and the length of each chromosome to 40 . The size of the population is a serious parameter. A size of subminiature weakens the method 's effectivenss. A whacking size presents the method slow. So the choice of population size is a critical task. We have done a lot of testing and found that values in the interval $[500, 2000]$ are proper. The length of the chromosomes is usually depended on the problem to be solved. In these experiments the maximum number of generations allowed was set to 500 and the preset fitness target for the termination criteria was 10^{-6}. In table 2 we list the results from the proposed method for the above examples. Under deadings MIN, MAX, AVG we list the minimum, maximum and average of generations in the set of 25 experiments.

Table 2. Method results for Examples.

OCP	MIN	MAX	AVG
$Exam4.1$	28	290	150
$Exam4.2$	25	300	175
$Exam4.3$	32	260	148
$Exam4.4$	15	275	136

8 Conclusion

We expressed a novel idea for solving FDE. The method is based on genetic programming. This idea creates trial solutions and seeks the best them. The advantage is that our method can produce analytical solutions that are approximate or even exact. If however the exact solution can not be represented in a closed form, our method will produce a closed form approximate.

References

1. S. Abbasbandy, Numerical methods for fuzzy differential inclusions, Computers and Mathematics with Applications, 48 (2004) 1633–1641.
2. S. Abbasbandy and T. AllahViranloo, Numerical solution of fuzzy differential equation by RungeKutta method, Nonlinear Studies, 11 (1) (2004) 117–129.
3. E. Babolian, H. Sadeghi, and Sh. Javadi, Numerically solution of fuzzy differential equations by Adomian method, Applied Mathematics and Computation, 149 (2004) 547–557.
4. B. Bede and S.G. Gal, Generalizations of the differentiability of fuzzy-number-valued functions with applications to fuzzy differential equations, Fuzzy Sets and Systems, 151 (2005) 581–599.
5. B. Bede, I.J. Rudas, and A.L. Bencsik, First order linear fuzzy differential equations under generalized differentiability, Information Sciences, 177 (2007) 1648–1662.
6. Y.C. Cano and H.R. Flores, On new solutions of fuzzy differential equations, Chaos, Solitons and Fractals, 38 (1) (2008) 112–119.
7. S.S.L. Chang and L. Zadeh, On fuzzy mapping and control, IEEE Transactions on System, Man and Cybernetics, 2 (1972) 30–34.
8. D. Dubois and H. Prade, Towards fuzzy differential calculus, Fuzzy Sets and Systems, 8 (1982) 225–233.
9. M. Friedman, M. Ma, and A. Kandel, Numerical solutions of fuzzy differential and integral equations, Fuzzy Sets and Systems, 106 (1999) 35–48.
10. R. Goetschel and W. Voxman, Elementary fuzzy calculus, Fuzzy Sets and Systems, 18 (1986) 31–43.
11. E. Hullermeier, Numerical methods for fuzzy initial value problems, International Journal of Uncertainty Fuzziness Knowledge-Based Systems, 7 (1999) 439–61.
12. O. Kaleva, Fuzzy differential equations, Fuzzy Sets and Systems, 24 (1987) 301–317.
13. J.R. Koza, Genetic Programming, On the Programming of Computer by Means of Natural Selection, MIT Press: Cambridge, MA, 1992.

14. M.T. Mizukoshi, L.C. Barros, Y. Chalco-Cano, H. Roman-Flores, and R.C. Bassanezi, Fuzzy differential equations and the extension principle, Information Sciences, 177 (2007) 3627–3635.
15. M. O'Neill, Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution, PhD thesis, University Of Limerick, Ireland, August, 2001.
16. M. O'Neill and C. Ryan, Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, volume 4 of Genetic programming. Kluwer Academic Publishers, 2003.
17. M. O'Neill and C. Ryan, Grammatical evolution, IEEE Trans. Evolutionary Computation, 5 (2001) 349–358.
18. M.L. Puri and D. Ralescu, Differential for fuzzy function, Journal of Mathematical Analysis and Applications, 91 (1983) 552–558.
19. C. Ryan, J. J. Collins, and M.O. Neill, Evolving programs for an arbitrary language, in Proceedings of the First European Workshop on Genetic Programming, Volume 1391 of LNCS, W. Banzhaf, Ri. Poli, M. Schoenauer and T.C. Fogarty, (eds.), Springer-Verlag, 8395, Paris, 1415 April 1998.
20. I.G. Tsoulos and I.E. Lagaris, Solving differential equations with genetic programming, 7 (2006) 33–54.
21. L.A. Zadeh, Toward a generalized theory of uncertainty (GTU) an outline, Information Sciences, 172 (2005) 1–40.
22. L.A. Zadeh, Fuzzy sets, Information and Control, 8 (1965) 338–353.