Solving of Ordinary differential equations with genetic programming

M.E. Wahed[1], Z.A. Abdelslam[2], O.M. Eldaken[2]

[1] Department of Computer Sciences, Faculty of computers and information, Faculty of Science, Suez Canal
[2]Department of Mathematics, Faculty of Science, Suez Canal University, Suez Canal, Egypt
osama_m_edaken@yahoo.com

Abstract: In this work, a novel hybrid method for the solutions of ordinary equations is presented here. The method creates trial solutions in genetic programming on grammatical evolution. The trial solutions are enhanced periodically using a local optimization procedure.
[M.E. Wahed, Z.A. Abdelslamand O.M. Eldaken **Solving of Ordinary differential equations with genetic programming.** *J Am Sci* 2015;11(8):12-16]. (ISSN: 1545-1003). http://www.jofamericanscience.org. 3

Keywords: Differential equations, genetic programming and grammatical evolution.

1.Introduction

A series of problems in many scientific fields can be modeled with the use of ordinary differential equations such as problems in physics [1–5], chemistry [6–8], biology [9, 10], economics [11], etc. The technique of genetic. Programming[2], is an optimization process based on the evolution of a large number of candidate solutions through genetic operations such as replication, crossover and mutation [12], These methods choose a basis set of functions with adjustable parameters and proceed approximating the solution by varying these parameters. Our method offers closed form solutions, however the variety of the basis functions involved is not a priori determined, rather is constructed dynamically as the solution procedure proceeds and can be of high complexity if required. This last feature is the one that distinguishes our method from others. We have not dealt with the problem of differential equation induction from data. The generation is achieved with the help of grammatical evolution. We used grammatical evolution instead the “classic "tree based genetic programming, because grammatical evolution can produce programs in an arbitrary language, the genetic operations such as crossover and mutation are faster and also because it is far more convenient to symbolically differentiate mathematical expressions. The code production is performed using a mapping process governed by a grammar expressed in Backus Nauru Form. Grammatical evolution has been applied successfully to problems such as symbolic regression [3],The rest of this article is organized as follows: in Section2 a brief description of the grammatical evolution algorithms given followed by analytical description of the proposed method. The test functions used in the experiments followed by the experimental results are outlined.

2 Method Description

In this section a brief description of the grammatical evolution algorithm is given**.** The main steps of the proposed algorithm are outlined with the steps for the fitness evaluation for the cases of ODEs.

2.1 Grammatical Evolution

Grammatical evolution is an evolutionary technique that can produce code in any programming language requiring the grammar of the target language in BNF syntax and some proper fitness function. This technique has been used with success in many scientific fields such as symbolic regression [13], by replacing non terminal symbols with the right hand of the selected production rule. The selection is performed in two steps:

- We read an element from the chromosome (with value ***V***).
- We select the rule according to the scheme

Rule = *V*mod NR(1)

Where NR 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. If the limit of the wrapping events is reached the chromosome is rejected by assigning to it a large fitness value, which prevents the chromosome to be used in the crossover procedure. In the proposed algorithm the limit of.wrapping.events was set to2.As an example of the mapping procedure of the grammatical evolution consider the BNF grammar shown in Fig. 1. The number in parent theses denotes these quince number of the corresponding production rule to be used in the mapping procedure. Consider the chromosome x=[9,8,7,6,16,10,17,23,8,14]. The steps of the mapping procedure are listed in Table1. The final outcome of these steps is the expression 3+sin(x).

2.2 Algorithm decription

The proposed method is baased on an evolutionary algorithm,a stochastic process whose basis lies in the biological evolution.

S: : = <expr> (0)
<expr>: : = (<expr> <op> <expr>) (0)
| (<expr>) (1)
| <func> (<expr>) (2)
| <digit> (3)
| x (4)
| y (5)
| z (6)
<op>: : = + (0)
| - (1)
| * (2)
| / (3)
<func >: : = sin (0)
| cos (1)
| exp (2)
| log (3)
<digit >: := 0 (0)
| 1 (1)
| 2 (2)
| 3 (3)
| 4 (4)
| 5 (5)
| 6 (6)
| 7 (7)
| 8 (8)
| 9 (9)

Fig.1 the grammar of theproposed method

Table 1: An example of the mapping procedure

string	chromosome	operation
<expr >	9, 8, 7,6,16,10,17,23,8,14	9 mod 7 = 2
<func>(< expr>)	8,7,6,16,10,17,23,8,14	8 mod 4 = 0
sin(<expr >)	7,6,16,10,17,23,8,14	7mod 7 = 0
sin(< expr> <op> <expr>)	6, 16,10,17,23,8,14	6 mod 7 = 6
sin(x <op> <expr>)	16,10,17,23,8,14	16 mod 4 = 0
sin(x) + <expr>	10, 17,23,8,14	10 mod 7 = 3
sin(x) + < digit>	17, 23,8,14	17 mod 10=7
sin(x) + 3	23,8,14	

Algorithm along with a penalty function which is used in order to represent the boundary or initial conditions of the ordinary differential equations, the main steps of the algorithm are as follows:

1. Set the number of chromosomes S, the number of maximum generations allowed K, the crossover rate pc, the mutation rate pm, a small positive number ε the integer parameter G and the integer parameter M. The parameter G determines show frequently the local search procedure will be applied and the parameter M determines inchworm any chromosomes the local optimization procedure will be applied.

2. Set inters = 0

3. Initialize.the chromosomes.Each chromosomes will be

4. Calculate the fitness for every chromosome

5. Apply the genetic operations of crossover and mutation to the population

2.3. Fitness evaluation

We express the ODEs in the following for

$$f(x,y,y^1,.....,y^{n-1},y^n) = 0\ ,\ x \in [a,b] \quad (2)$$

where y^n denotes the n-order derivative of y let the boundary or initial conditions be given by:

$$\psi_{i(x,y,y^{(1)},.....,y^{(n-1)},y^{(n)})}\Big|_{x=t_i} = 0 \quad \text{i=1,..., n} \quad (3)$$

where t_i is either a or b: The steps for the fitness evaluation of any given chromosome g are:

1. Choose T equidistant points in [a; b] denoted by $[x_0,x_1,x_2,....,x_T]$

2. For every chromosome i

(a) Construct the corresponding model $g_i(x)$ expressed in the grammar described earlier

(b) Calculate the quantity

$$E(g_i) = \sum_{j=0}^{T-1}(f(x_j,g_i^{(0)}(x_j),....g_i^{(n)}(x_j) \quad (4)$$

(c) Calculate an associated penalty $P(g_i)$ as shown below.

(d) Calculate the fitness value of the chromosome as:

$$\Phi_i = E(g_i) + P(g_i) \quad (5)$$

The penalty function P depends on the boundary conditions and it has the form:

$$P(g_i) = \lambda\sum_{k=1}^{n}\psi_k{}^2(x,g_i,g_i^{(1)},.....,g_i^{(n-1)}) - (6)$$

where λ is a positive number

3 Experiments

ODE1:

$$\frac{dy}{dx} = \frac{2x-y}{x}$$

with $y(1) = 3$ and $x \in [1,3]$ the analytical solution is $y(x) = x + \frac{2}{x}$

ODE2:

$$x\frac{dy}{dx} + 2y = x^2 - x + 1$$

with $y(1) = \frac{1}{2}$ and $x \in [1,3]$ the analytical solution

$$y(x) = \frac{1}{4}x^2 - \frac{1}{3}x + \frac{1}{2} + \frac{1}{12x^2}$$

ODE3:

$$\frac{dy}{dx} = \frac{1-y\cos(x)}{\sin(x)}$$

with $y(1) = \frac{3}{\sin(1)}$ and $x \in [1,3]$ the analytical solution is $y(x) = \frac{x+2}{\sin(x)}$

3.2 Experimental Results

The method was performed 30 times, using different seeds for the random number generator each

time, on every ordinary differential equation described previously and averages were taken. In Table 2 the numerical values for the parameters of the algorithm are listed. The local optimization procedure used in the experiments was a BFGS variant due to Powell [14]. In Table 3 compare between the results from the application the exact solution,the error calculated at range $x \in [1,3]$.

In Fig.2, The application of the final solution in range [1:3] is plotted against the true solution $y(x) = x + \frac{2}{x}$ At generation 22 the fitness value was 4200 and the intermediate solution was:

$$y(x) = -\sin(7)\sin\left(\cos\left(\cos\left(\sin(7)\, e^{\cos(6)-5}\right)\right)\sin\left(2x\ln(2)\right)\right) + \frac{e}{sin\,(1)}$$

In Fig.3, The application of the final solution in range [1:3] is plotted against the true solution $y(x) = \frac{1}{4}x^2 - \frac{1}{3}x + \frac{1}{2} + \frac{1}{12x^2}$

At generation 35 the fitness value was 6140 and the intermediate solution
was:

$$y(x) = \frac{34(x(\ln(x) + \sin(\ln(3))) + 1)}{81\ln(x) + 54\sin(\ln(3))}$$

In Fig.4, The application of the final solution in range [1:3] is plotted against the true solution $y(x) = \frac{x+2}{\sin(x)}$

At generation 20 the fitness value was 4750 and the intermediate solution
was:

$$y(x) = \frac{124}{x}\left(\frac{x^3\sin(2)}{e^2} + \sin(x)\right) - 113$$

Table 2: the numerical values for the parameters of the method

Name	Value
S	500
K	2000
P_c	0.9
P_m	0.05
ε	10^{-6}
λ	100
G	20
M	20
T	100
B	10

Table3: Experimental results

Problem	Exact	G.P.	Error
ODE1			
X=1	3	2.8976	0.1324
X=1.5	2.8667	2.9317	-0.065
X=2	3	3.1032	-0.1032
X=2.5	3.3	3.3429	-0.0429
X=3	3.6667	3.5219	0.1448
ODE2			
X=1	0.5	0.4689	0.0311
X=1.5	0.5995	0.7302	-0.1307
X=2	0.8542	1.0336	-0.1794
X=2.5	1.2425	1.3684	-0.1259
X=3	1.7593	1.7282	0.0311
ODE3			
X=1	3.5652	6.6	-3.0348
X=1.5	3.5088	3.794	-0.2852
X=2	4.3988	4.42	-0.0212
X=2.5	7.5191	12.056	-4.6572
X=3	35.431	30.16	5.271

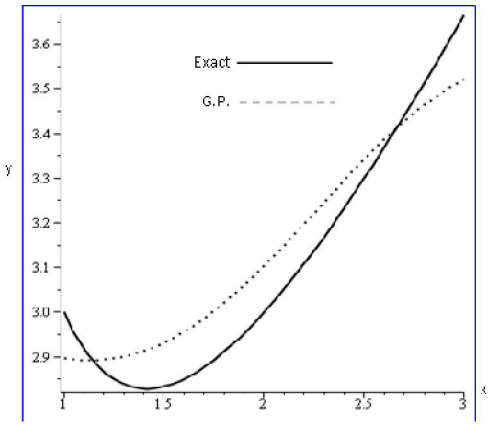

Fig. 2: G.P. and Exact solutions for ODE 1

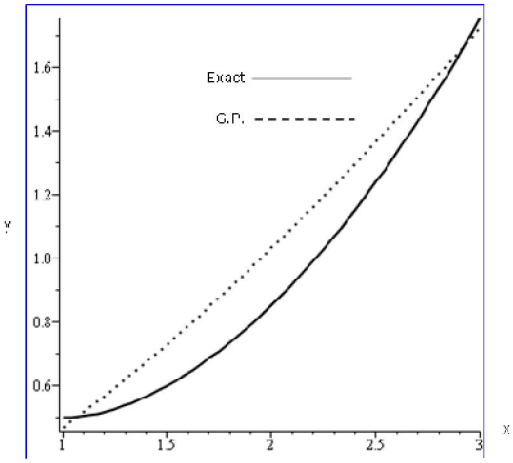

Fig. 3:G.P. and Exact solutions for ODE 2

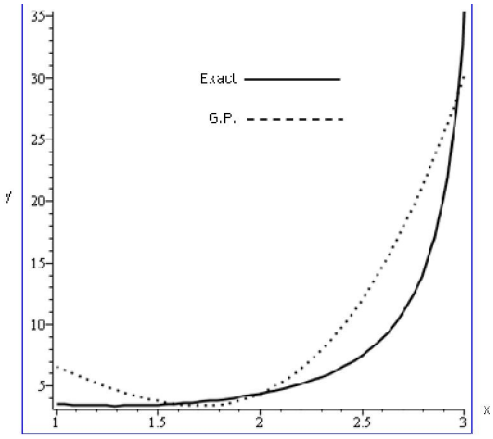

Fig. 4:G.P. and Exact solutions for ODE 3

4 Conclusions

In this Section, we represent the obtained Genetic programming results for the ODEs problems, and comparison between Genetic programming method and exact solution and calculate the error between it. Fig. 2, Fig. 3, Fig. 4 and table 3 show that the values of error better in the case of functions which have small values. In the future we will try to improve this way when the values are big. Although the error is relatively large in some regions, the advantage of this method for numerical methods is finding a function which give approximate solution for ODEs, and can be handled with this function and study its behavior so this is better in some boundary value problems from numerical methods.

References

1. Its A.R., A.G. Izergin, V.E. Korepin,N.A. Slavnov, Differential equations for quantum. correlation functions, International Journal of Modern Physics B4 (1990):1003–1037.
2. Kotikov A.V., Differential equations method: the calculation of vertex-type Feynman diagrams, Physics Letters B259(1991):314–322.
3. Gang H., H. Kaifen, Controlling chaos in systems described by partial differential equations, Phys. Rev. Lett. 71(1993):3794–3797.
4. Budd C.J., A. Iserles, Geometric integration: numerical solution of differential equations on manifolds, philosophical transactions: mathematical, Physical and Engineering Sciences: Mathematical 357(1999):945–956.
5. Peng Y.Z., Exact solutions for some nonlinear partial differential equations, Physics Letters A314(2003):401–408.
6. Verwer J. G., J. G. Blom, M. vanLoon, E. J. Spee, A comparison of stiff ODE solvers for atmospheric chemistry problems, Atmospheric Environment 30(1996):49–58.
7. Behlke J., O. Ristau, A new approximate whole boundary solution of the Lamm differential equation for the analysis of sedimentation velocity experiments, Biophysical Chemistry 95(2002):59–68.
8. Salzner U., P. Otto, J. Ladik, Numerical solution of a partial differential equation system describing chemical kinetics and diffusion in a cell with the aid of compartmentalization, Journal of Computational Chemistry 11(1990):194–204.
9. Koza J. R., Genetic Programming: On the Programming of Computer by Means of Natural Selection. MIT Press: Cambridge, MA, 1992.
10. O'Neill M. and C. Ryan, Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language, volume 4 of Genetic programming. Kluwer Academic Publishers, 2003.
11. Collins J. J. and C. Ryan, "Automatic generation of robot behaviors using grammatical evolution," in Proc. of AROB 2000, the Fifth International Symposium on Artificial Life and Robotics.
12. Goldberg D. E., Genetic algorithms in search, Optimization and Machine Learning, Addison Wesley, 1989.
13. O'Neill M., C. Ryan, Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language, Genetic Programming, vol.4, Kluwer Academic Publishers, Dordrecht, 2003.
14. Powell M. J. D., A tolerant algorithm for linearly constrained optimization calculations, Mathematical Programming 45(1989):547–566.