Global Journal of Researches in Engineering: J
General Engineering
Volume 22 Issue 2 Version 1.0 Year 2022
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 2249-4596 & Print ISSN: 0975-5861

Applications of Genetic Programming and Automatic Differentiation Algorithms in the Solution of Ordinary and Partial Differential Equations

By W. J. A. Lobão, M. A. C. Pacheco, D. M. Dias & A. C. A. Abreu
Pontifical Catholica University

Abstract- There is a significant number of research projects using differential equations to model important and complex problems of engineering and other scientific knowledge areas. This paper investigates the potential that computational algorithms have to determine analytical solutions for ordinary and partial differential equations. In order to do so, the evolutionary method of genetic programming and the automatic differentiation method are applied. Using the MatLab programming environment, several GPAD algorithms are developed and problems of distinct differential equations are addressed. The results are promising, with exact solutions obtained for most of the addressed equations, including ones that commercial systems could not find a symbolic solution to. The conclusion is that GPAD algorithms can be used to discover analytic solutions for ordinary differential equations and partial differential equations.

Keywords: evolutionary algorithm; genetic programming; automatic differentiation; differential equations.

GJRE-J Classification: DDC Code: 515.353 LCC Code: QA379

APPLICATIONSOFGENETICPROGRAMMINGANDAUTOMATICDIFFERENTIATIONALGORITHMSINTHESOLUTIONOFORDINARYANDPARTIALDIFFERENTIALEQUATIONS

Applications of Genetic Programming and Automatic Differentiation Algorithms in the Solution of Ordinary and Partial Differential Equations

W. J. A. Lobão [α], M. A. C. Pacheco [σ], D. M. Dias [ρ] & A. C. A. Abreu [ω]

Abstract- There is a significant number of research projects using differential equations to model important and complex problems of engineering and other scientific knowledge areas. This paper investigates the potential that computational algorithms have to determine analytical solutions for ordinary and partial differential equations. In order to do so, the evolutionary method of genetic programming and the automatic differentiation method are applied. Using the MatLab programming environment, several GPAD algorithms are developed and problems of distinct differential equations are addressed. The results are promising, with exact solutions obtained for most of the addressed equations, including ones that commercial systems could not find a symbolic solution to. The conclusion is that GPAD algorithms can be used to discover analytic solutions for ordinary differential equations and partial differential equations.

Keywords: evolutionary algorithm; genetic programming; automatic differentiation; differential equations.

I. Introduction

A significant number of research projects, in different areas of scientific knowledge, use mathematical models partially or fully formulated by differential equations. However, given the complexity of the proposed models, difficultlies with problems and unknown analytic solutions often arise. In this instance, the solution is often sought throught the use of numerical methods.

The majority of these projects aim at achieving better results by using differential equations in order to try to describe the dynamic behavior of variables. In those cases, the numerical solutions are usually not complete. A complete solution form, however, is necessary because it allows for important and different types of analyzes, such as comparative static, knowledge of the magnitude of the partial effects, calculation of elasticities and studies on stability and stationarity.

The primary objective of this paper is to investigate the potential of evolutionary algorithms to obtain analytic solutions to ordinary and partial differential equations (ODE) and (PDE). In order to do that, the evolutionary algorithms were developed by using the MatLab and combining genetic programming (GP) and automatic differentiation (AD). Several problems with different kinds of ODE and PDE were used for testing. More than twenty equations with analytic solutions known from literature were analyzed. An equation with an uknown analytic solution deriving from a PDE, which describes the wave function of the Schrödinger equation for helium atom, was also used. This equations is shown in [1]. The results of the tests are promising, since we obtained exact solutions for all equations with known solutions while identically re-producing the existing solutions. In the case of the Schrödinger equation, the solution obtained for this PDE is approximated. Hence, it has a low error margin.

To compare the developed methodology with the existing ones, we underwent a bibliographic review. It was verified that, although an extended amount of literature on genetic programming and automatic differential exists, the majority of the works apply these methods separately. Furthermore, amongst the works that develop combined algorithms of GP and AD, namely GPAD, papers that aim at solving differential equations are rare. Among these, the articles [2–4] stand out.

In the work of Imae et al [2], a GPAD algorithm is proposed to solve Hamilton-Jacobi-Bellman, Hamilton-Jacobi-Isaacs and Francis-Byrnes-Isidori equations in dy-namic nonlinear systems with optimal control problems. The methodology uses the conventional GP technique, combined with the AD method, developed by the authors themselves. The algorithm is applied to equations with already known symbolic solu-tions and generates approximate solutions with low estimate errors, for every simulated case. The article's conclusion is that the methodology presents promising results and is a good alternative for solving problems of optimal control.

Author α: National School of Statistical Sciences of IBGE, Rio de Janeiro, Brazil. e-mail: waldir.lobao@ibge.gov.br
Author σ: Pontifical Catholica University of Rio de Janeiro, Rio de Janeiro, Brazil. e-mail: marco@ele.puc-rio.br
Author ρ: Faculty of Engineering at UERJ, Rio de Janeiro, Brazil. e-mail: douglas.dias@uerj.br
Author ω: e-mail: carolina@ele.puc-rio.br

In the work of [5], the two methods are combined to solve ordinary differential equations, with solutions that are already known. The equation of the Simple Harmonic Oscillator, of the mass-spring system, and Schrödinger equation for the hydrogen atom are used to obtain the exact solutions for the two proposed equations. The author concludes the study affirming that the combination of GP with AD is of great utility for the resolution of differential equations.

Among the researched articles, Tsoulos and Lagaris [6] present the most diverse and highest number of solved examples for differential equations using GPAD algorithms. Solutions are presented for ordinary differential equations (ODEs) and partial (PDEs), linear and nonlinear, of first and second degree. Exact symbolic solutions are found for the vast majority of the addressed problems. The developed algorithm uses gram-matical evolution for the GP algorithm and different AD methods. The conclusions are that the developed methodology is able to solve ODEs and PDEs and the results are very encouraging.

These works served as the basis of comparison to evaluate the performance of the algorithms developed and were of fundamental importance in the search for better results and the improvement of the methodology created. In addition, we have previously adopted a methodology, present in [7], that also combined genetic programming and automatic differentiation, to obtain symbolic solutions to stochastic differential equations. The results achieved were promising and indicated the effectiveness of the proposed methodology in solving such equations and modeling problems that involve stochastic differential equations. Similarly, our results in this article are encouraging, as exact solutions were achieved for the vast majority of the addressed problems. In addition, they indicate that the proposed method can be an efficient alternative to solve ODE and PDE.

This article is organized in three sections. The first section consists of a review of previous literature. The second section describes the methodological procedures adopted in the development of the GPAD algorithms. The third section discusses the application results of the proposed methodology and presents the solution for the ODEs and PDEs. The fourth and final section summarizes and concludes the article.

II. Methodological Aspects

This section introduces the methodological procedures adopted for developing the GPAD algorithm, elaborated with the purpose of solving problems of differential equations. Its description of the methodological aspects related to the structure and operation of the algorithm steps are limited. See [4, 8–10] for technical details about the genetic programming and automatic differentiation methods used in this article.

a) Automatic Differentiation

Automatic differentiation (AD) is a set of techniques based on the mechanical application of the chain rule designed to numerically evaluate the derivative of a function specified by a computer program. It exploits the fact that every computer program, no matter how complicated, executes a sequence of elementary arithmetic operations (addition, subtraction, multiplication, division, etc.) and elementary functions (exp, log, sin, cos, etc.). Derivatives of arbitrary order can be computed automatically by applying the chain rule repeatedly to these operations. This way, they accurately work precision and use more arithmetic operations than the original program.

The classical methods, however, are not bereft of problems. Symbolic differentiation often leads to relatively inefficient codes and difficulties converting a computer program into a single expression. Meanwhile, numerical differentiation can introduce round-off errors in the discretization process and cancellation. Both classical methods have problems with calculating higher derivatives, which cause the complexity and errors to increase. Furthermore, both classical methods are slow at computing the partial derivatives of a function of many inputs, as is needed for gradient-based or stochastic optimization algorithms. AD circumvents these problems at the expense of introducing more software dependencies.

Usually, two distinct modes of AD are presented: forward mode, or forward accumu-lation, and reverse mode, or reverse accumulation. Forward mode specifies that chain rule can be transvered from inside to outside. Since for the simple composition $y = g\,(h(x)) = g\,(w)$ the chain rule gives $dy/dx = (dy/dw)\,(dw/dx)$, the forward mode first computes (dw/dx) and then (dy/dw), while the reverse mode has the traversal from outside to inside. See [11] for details.

b) GPDA Algorithm

The algorithm is developed in MatLab programming environment and works with two basic codes operating simultaneously. The first is responsible for the implemen-tation of genetic programming, and the second performs the automatic differentiation and evaluates the individuals' fitness. The codes run a fixed number of generations and perform the following steps: the creation of initial population, fitness evaluation of individuals, selection of individuals for reproduction, reproduction and validation, survival and creation of the new population. Except for the first step, which runs only at initialization, the remaining steps are repeated in all generations.

The form of representation of individuals is the traditional tree diagram, which is appropriated to evolve equations and facilitates to interpret GPAD results. The GP parameterization is flexible and allows changes to the configuration of parameters set. Some of these parameters are: types and forms of trees, minimum depth of initial population trees, maximum depth of trees, the number of nodes control, set of functions and terminals, number of generations, selection methods for reproduction, crossover and mutation rates, population size, and stop conditions.

Two essential components for the GP operation are the sets of functions and terminals. These are necessary for the creation and reproduction of the population of individuals and, consequently, are involved directly in the formation of the optimal solution to the problem. The set of functions is composed of basic mathematical operators $(+,-,\times,\div)$ and elementary functions (e.g., $sin, cos, exp, log, xy, tan$). The terminal set consists of constants (e.g., real numbers, complex numbers, number, and randomly generated numbers) and variables that constitute the differential equation.

The choice of both functions and constants sets is important for the GP performance. If these sets are chosen with a small number of elements, the algorithm will present premature convergence due to the lack of diversity and a acceptable solution to the problem will not be reached. On the other hand, if they are chosen with an excessive number of elements, the search space becomes very large and harms the de-termination of an acceptable solution, which also increases the computational effort. Therefore, the choice procedure must be accomplished in a balanced way, respecting the properties of closure and sufficiency, as defined by [9]. The strategy used in this paper, which has yielded good results, was to mathematically study the problem of in-terest, ODE or PDE, and identify the minimal set of functions and constants required for their solution. Then, the algorithm was started with the basic set and new func-tions and constants were added to the modelling process until a satisfactory solution was obtained.

The first step that GP undergoes is creating the initial population. Functions and terminals are selected and combined, originating the individuals that make up the initial population. In this study, each created individual is a mathematical expression represented as a tree and possibly a solution of the differential equation of interest. GPAD offers three methods for the selection of the initial population: Full, Grow, and Ramped-half-and-half. The Grow method was used in all applications on this paper because it showed more efficiency in the processing time and produced the best results.

The second step consists of a fitness evaluation of the individuals. An error measure (i.e., fitness) was defined to evaluate the performance of each individual. After ana-lyzing the measures, it was decided to work with mean absolute error (MAE) between the differential equation derived from the function proposed by GP and the differential equation of the problem in question. We also regarded penalties for missed restrictions such as initial and boundary conditions. In other words:

$$fitness = MAE + restrictions\_error \quad (1)$$

It was also necessary to incorporate a code into the GPAD algorithm that performs differential calculus and is able to calculate the fitness. The AD method was chosen because it calculates the exact values of the derivatives of a function for a given set of input values.

The AD program has the task of mathematically verifying the solutions proposed by the GP program. For this purpose, the AD programs my AD and my A2D developed by [8] were used. This programs perform the first and second order derivatives of a function, respectively, and operate in a forward mode. After adjusting the AD codes to make them compatible with the GP code, they were incorporated into the GPAD. An example is provided to illustrate this step using the following ODE problem:

$$f'(x) + 2f(x) = exp(-2x);\ x \epsilon R, initial\ conditionl\ f(0) = 3 \quad (2)$$

Suppose that the function below was newly created by an individual GP and is a possible solution to the problem.

$$f(x) = 2 + exp(-2x) \quad (3)$$

The next step is to evaluate the fitness of equation (3). The second code performs automatic differentiation and calculates the derivative of f(x) at each point of the domain defined by the algorithm (usually a grid with 50 or more points of x), reaching the solution:

$$f^{'}(x) = -2exp(-2x) \quad (4)$$

Eequations (3) and (4) are placed within equation (2) and the proposed solution fitness is evaluated at each point. The symbolic expression of this comparison is:

$$f^{'}(x) + 2f(x) - exp(-2x) = 4 - exp(-2x) \neq 0 \quad (5)$$

f(x) is not the exact solution to equation (2) because its error value is not zero.
However, it satisfies the initial condition, since:

$$f(0) = 2 + exp(-2 \times 0) = 2 + 1 = 3 \quad (6)$$

Global Journal of Researches in Engineering (J) Volume XXII Issue II Version I Year 2022

Moreover, the algorithm does not work with symbolic differentiation, as shown before, but with the automatic differentiation, where the derivatives are applied and evaluated at every point of the domain of the function. The error measure (fitness) for this example is:

$$fitness = (\frac{1}{n}) + \sum | f'xi + 2f(xi) - |2xi| + |f(0) - 3| \quad (7)$$

Where the first part is the MAE, calculated in n domain points considered, and the second part is the penalty for not satisfying the initial condition of the problem.

Once the fitness function is defined, the GP evaluates the newly created individu-als and selects those who are best fit to take part in the reproduction process. The algorithm provides five selection methods for reproduction: roulette, sus, tournament, lexictour and doubletour. For the purpose of this paper, the lexictour is often used, as it generates better results than others, possibly due to its use of lexicographic parsimony pressure[3].

Once the GP finishes the selection, it initiates the individuals' reproduction. This step is of fundamental importance as it leads the GP into regions with a better search space and, consequently, improves the process of optimization. Once the application of crossover and mutation operators takes place, the process of reproduction of the individuals is concluded. The new individuals are subjected to a validation test. The GP considers an individual fit to participate in the new population if its tree size does not exceed the maximum size parameter value.

In the last step, the GP selects the individuals that survive to be a part of the next population. The algorithm provides four elitism methods: replace, keepbest, halfe-litism, totalelitism. Totalelitism was used in this research, as it allows all individuals to participate in the selection and selects the fittest ones, regardless of their parent-age. Individuals are selected until the population reaches its predefined size. The new population is, then, ready to be evaluated.

The process repeats itself until it reaches the stop condition and the final solution becomes known. The algorithm's overall objective is to create an individual (i.e., the solution f(x)) that minimizes the fitness, using the GP and AD. The search for the optimal solution happens in an evolutionary way and its determination depends on other technical procedures and refinements not shown in this paper.

Table 1: GPAD Basic Parameters

Parameter	Setup	Parameter	Setup
Number of generations	25 to 50	Initialization	growinit
Population size	50 to 600	Selection method	lexictour
Number of functions	8 to 15	Crossover	one-point
Number of terminals	5 to 15	Mutation	standart
AD Algorithm(fixed)	forward	Reproduction	totalelitism

The last step performed by the methodology is the validation of the best solution generated by the GPAD algorithm. This step occurs outside the programming environment. A solution will only be considered final if it meets the following requirements:

- fitness value $< 10^{-8}$;
- satisfying the restrictions and conditions laid down by the problem;
- when evaluated by the differential calculus rules, it must present a mathematical expression identical to the differential equation originally proposed.

If a final solution, validated by the requirements above, achieves fitness equal to zero, it will be considered an exact solution to the proposed problem. Otherwise, this solution is considered an approximate solution.

III. Applications and Results

This section presents the results of four examples, two ODE and two PDE, which illustrate how solutions of differential equations can be achieved through the use of GPAD.

In General, the GPAD was set up according to the parameters listed in table 1, for all the examples. The numeric parameters - related to the number of generations, individuals, elementary functions, and variables - were displayed in intervals of integers, as they varied according to the size and complexity of the addressed problem. The forward method was used for the application of the AD. This will be further explained in section 2.2. In all examples, the solutions obtained by the GPAD was represented in three different ways: symbolic, tree, and graphics.

a) Example 1

The first problem includes the solution of the following ODE of the second order:

$$y''(x) + y'(x) + y(x) = -1.6 + 0.4x + 0.2x^2 - 8sin(4x) - 30cos(4x);$$
$$where\ x \in \kappa, y''(x) = \frac{d^2y(x)}{dx^2}, y'(x) = \frac{dy(x)}{dx}; \quad (8)$$
$$y(0) = 0\ and\ y'(0) = 0$$

The solution obtained by the GPAD is presented in function (9) and in figures 1 and 2.

$$y(x) = 2 + 0.2x^2 + 2cos(4x) \quad (9)$$

Figure 1 shows the tree representation of the solution to the ODE of example 1 (3.1), where variable (x) is denoted by (X1). The optimal solution was found in individual 5054 (best so far), for a population of 250 individuals and 60 generations. The tree had a low number of depth and nodes, which made the reading and interpretation of the function easier. The achieved solution was exact, because it presented fitness equal to zero, satisfied all the constraints defined in (8), and was identical to the ODE of the problem proposed applying the differential calculus. Figure 1 is displayed below, along with a list of the results.

- Generations: 60
- Individuals: 250
- Best so far: 5054
- Fitness: 0.00000
- Depth: 5
- Nodes: 15

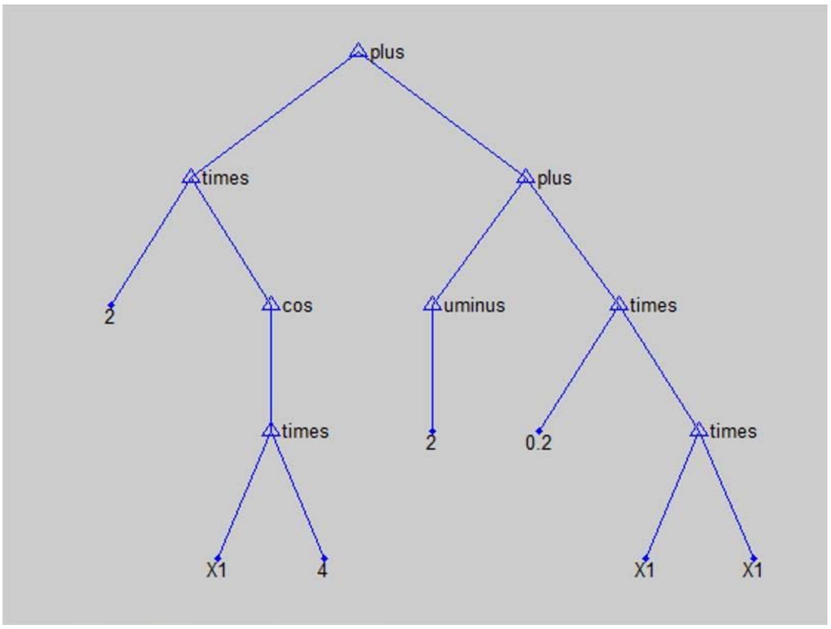

Figure 1: Tree representation of y(x), solution to the ODE of example 1.

Global Journal of Researches in Engineering (J) Volume XXII Issue II Version I Year 2022

Figure 2 shows the graph of solution (9):

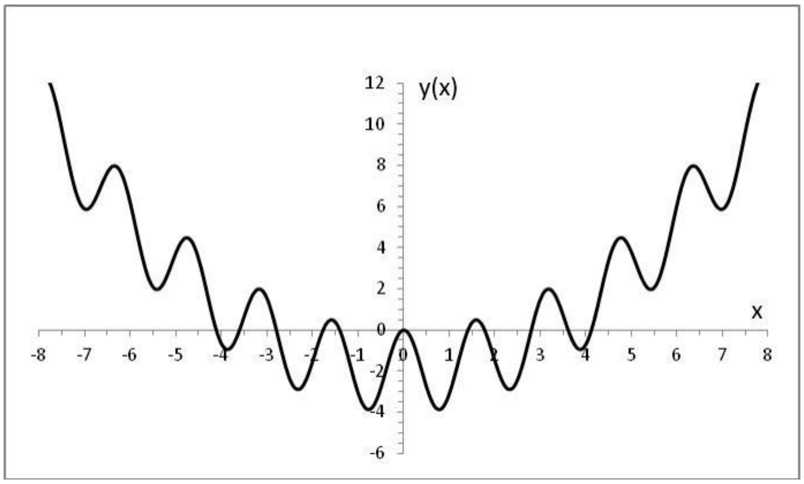

Figure 2: Graph of function y(x), solution to the ODE of example 1.

b) Example 2

The second problem contains the solution to the following ODE of second order, along with its constant coefficients:

$$y''(x) + 0.3y'(x) + 25y(x) = 25.12 + 10x - 1.5\,cos(5x)exp(-0.3x);$$

$$where x\epsilon\kappa;\ y(0) = 1\ and y'(0) = 5.4 \quad (10)$$

The solution obtained by the GPAD is presented in function (11) and in the figures 1 and 2.

$$y(x) = 1 + 0.4x + sin(5x)exp(-0.3x) \quad (11)$$

Figure 3 presents a graph which shows the tree representation of the solution of the ODE in example 2, where variable (x) is denoted by (X1). The optimal solution was found in individual 6031 (best so far), for a population of 400 individuals and 40 generations. The tree had a low numbers of depth and of nodes. The solution obtained was exact and the answer was verified by the usual differential calculus. Figure 3, along with the results, is displayed below.

- Generations: 40
- Individuals: 400
- Best so far: 6031
- Fitness: 0.00000
- Depth: 6
- Nodes: 16

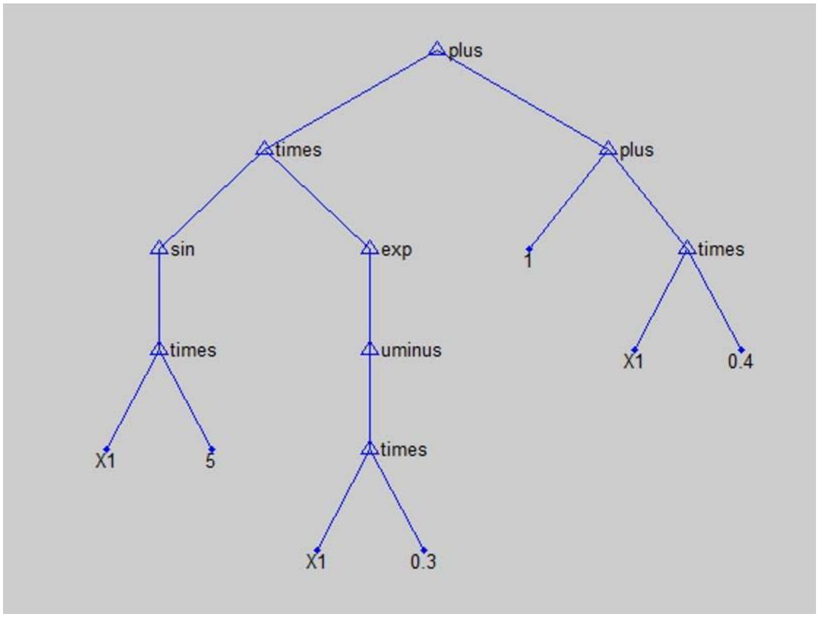

Figure 3: Tree representation of y(x), solution to the ODE of example 2.

Figure 4 contains a graph of solution (11), which exhibits non-linear and complex behavior.

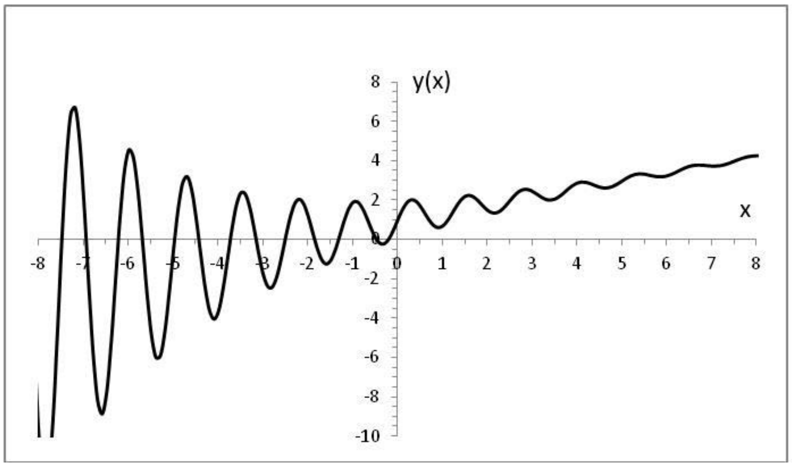

Figure 4: Graph of function y(x), solution to the ODE of example 2.

c) Example 3

The third equation contains the solution to the following ODE of the second order:

$$x^2y''(x) + xy'(x) + 4y(x) = 6sin(ln(x))$$

$$where \in \kappa, x > 0;\ y(1) = 3\ and\ y^{'}(1) = 2 \qquad (12)$$

The solution obtained by the GPAD for equation (12) was:

$$y(x) = 3cos\big(2ln(x)\big) + 2sin\big(ln(x)\big) \qquad (13)$$

Figure 5 shows a tree representation for the ODE solution in example 3, where (X1) denotes variable (x). The GPAD found the exact solution after 13,665 evaluations (best so far), with a population of 400 individuals and 50 generations. The tree presented depth 6 and 13 nodes. Figure 5, along with the results acquired, is displayed below.

- Generations: 50
- Individuals: 400
- Best so far: 13665
- Fitness: 0.00000
- Depth: 6
- Nodes: 13

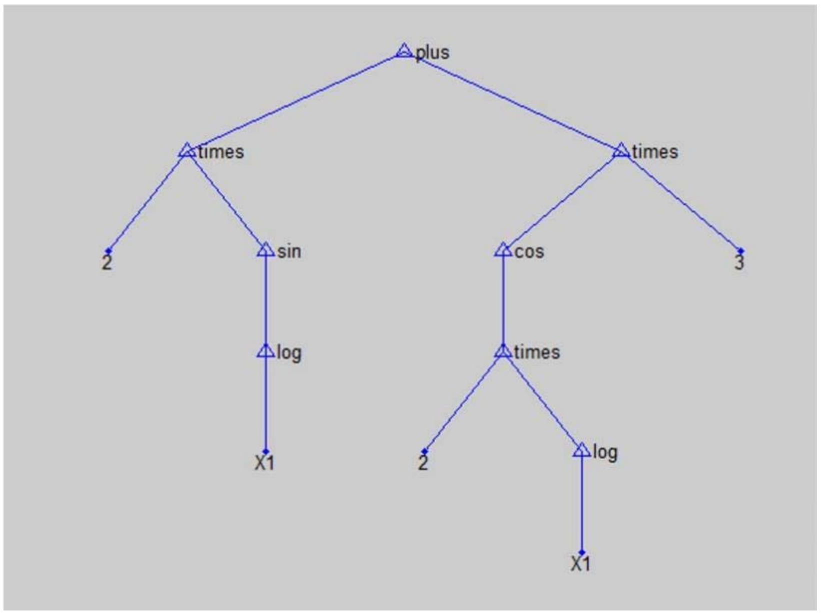

Figure 5: The representation of y(x), solution to the ODE of example 3.

Figure 6 shows the graph of solution y(x), in function (13).

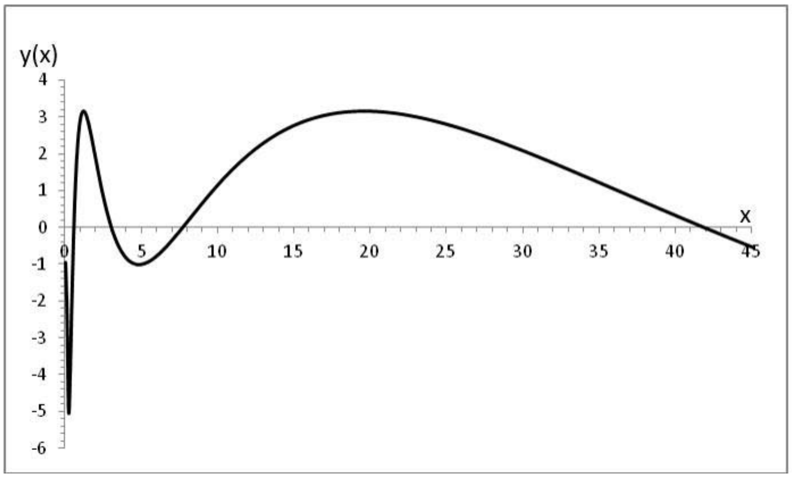

Figure 6: Graph of function y(x), solution to the ODE of example 3.

d) Example 4

The fourth application solves the following second order PDE, with variable coefficients being:

$$x f_x + 2 f_t = xt\cos(x) + 2\sin(x);$$
$$where\ f_x = \partial f(x,t)/\partial x, f\ t = \partial f(x,t)/\partial t, x\, \epsilon\, \kappa, t > 0; \quad (14)$$
$$f(0,t) = 0\ and\ f(x,0) = x^2$$

The solution obtained by GPAD to equation (14) is:

$$f(x,t) = x^2 exp(-t) + sin(x) \quad (15)$$

Figure 7 shows a tree representation for the PDE solution in example 4, where (X1) and (X2) denote variables (x) and (t), respectively. The GPAD found the exact solution after 4,823 evaluations (best so far), with a population of 600 individuals and only 25 generations.

The tree has depth 6 and 12 nodes. Figure 7, along with the data acquired, is displayed below.

- Generations: 25
- Individuals: 600
- Best so far: 4823
- Fitness: 0.00000
- Depth: 6
- Nodes: 12

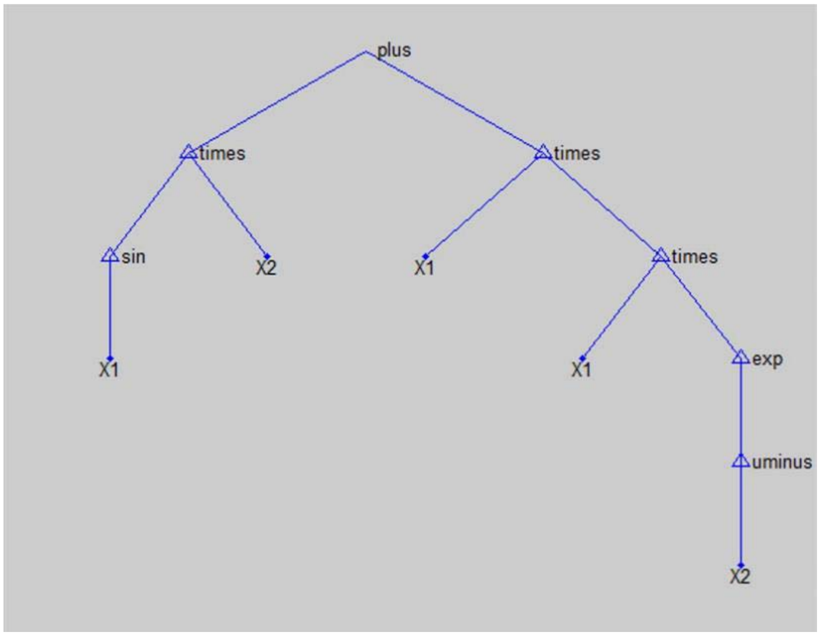

Figure 7: Tree representation of f(x,t), solution to the ODE of example 4.

Figure 8 shows a graph of solution f(x,t) in function (15).

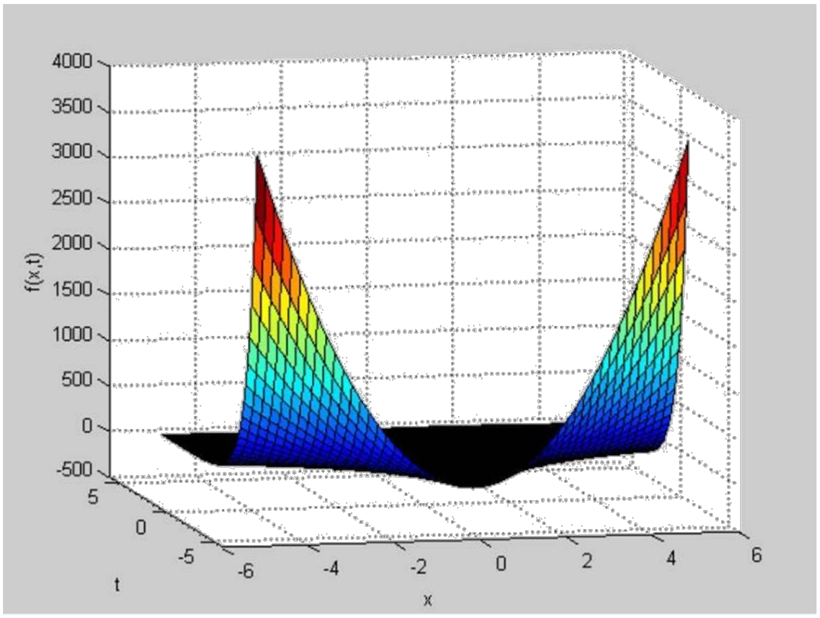

Figure 8: Graph of function f(x,t), solution to the ODE of example 4.

e) Example 5

The fifth example solves the following second order PDE, which is a heat equation particular case:

$$ft - fxx = [x\cos(t) - 4(x-1)\sin(t)]\exp(-2x);$$
$$where\ fxx = \partial^2 f(x,t)/\partial x^2, ft = \partial f(x,t)/\partial t, 0 \le x \le 1, 0 \le t \le 1; \qquad (16)$$
$$f(0,t) = 0, f(x,0) = 0\ and\ f_x(1,t) + f(1,t) = 0$$

The solution obtained by the GPAD to equation (16) is:

$$f(x,t) = \sin(t)x\exp(-2x) \qquad (17)$$

Figure 9 shows a tree representation for the PDE solution in example 5, where (X1) and (X2) denote variables x and t, respectively. GPAD found the exact solution after 2,054 evaluations (best so far), with a population of 80 individuals and 50 generations. The tree contains depth 6 and 10 nodes. Figure 10 shows the graph of solution f(x, t) in function (17). The gathered data is displayed below, along with figure 9.

- Generations: 25
- Individuals: 600
- Best so far: 4823
- Fitness: 0.00000
- Depth: 6
- Nodes: 12

Global Journal of Researches in Engineering (J) Volume XXII Issue II Version I Year 2022

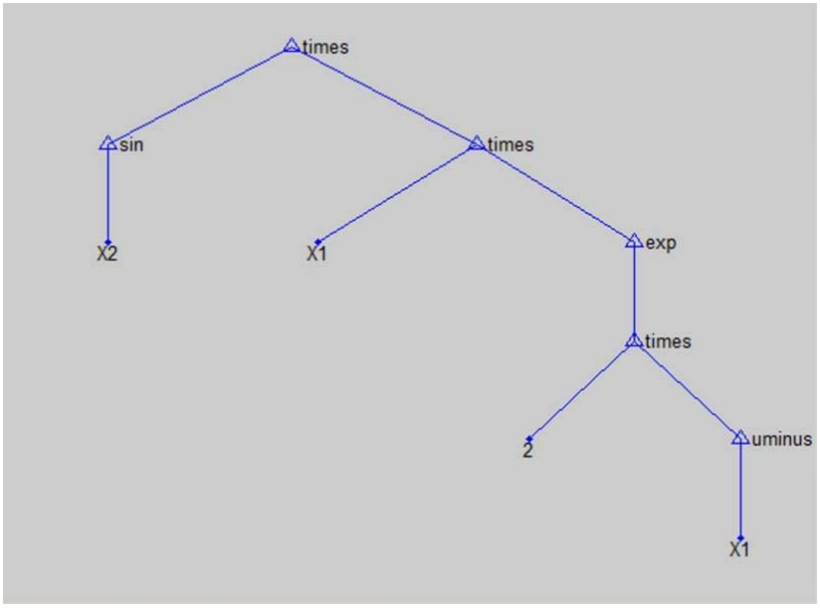

Figure 9: Tree representation of f(x,t), solution to the ODE of example 5.

Figure 10 shows the graph of solution f(x, t) in function (17).

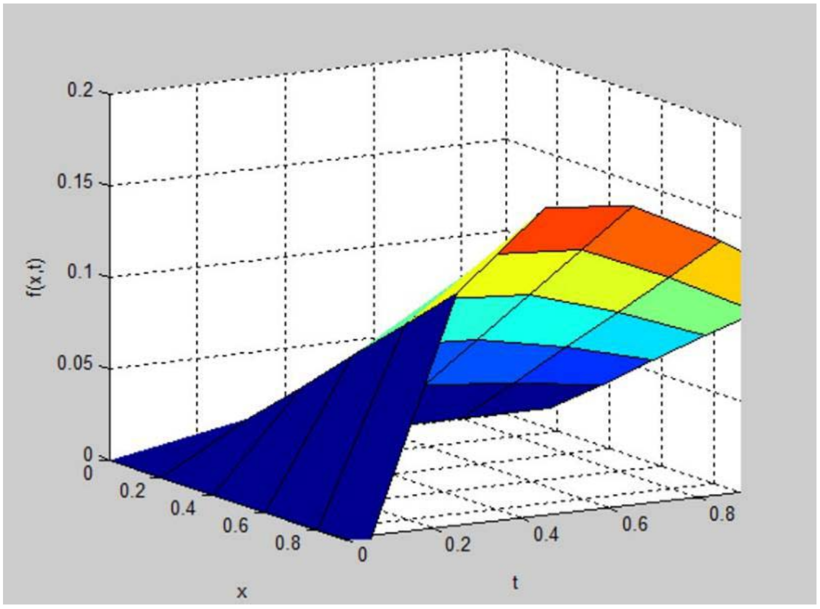

Figure 10: Graph of function f(x,t), solution of the ODE of example 5.

f) Example 6

The sixth and last example solves the following PDE:

$$f_t - f_{xx} = [x - t(4x^3 - 6x)\,exp(-x^2)];$$
$$where\ f_{xx} = \partial^2 f(x,t)/\partial x^2, ft = \partial f(x,t)/\partial t, 0 \le x \le 1, 0 \le t \le 1; \quad (18)$$
$$f(0,t) = 0, f(x,0) = 0\ and\ f_x(1,t) + f(1,t) = 0$$

The solution obtained by the GPAD to equation (18) is:

$$f(x,t) = xt\,exp(-x^2) \quad (19)$$

Figure 11 shows a tree representation for the PDE solution in example 6, where (X1) and (X2) denote variables x and t, respectively. The GPAD found the exact solution after 1,336 evaluations (best so far), with a population of 50 individuals and 50 generations. The tree has depth 5 and 9 nodes. The solution for the problem is shown below, along with figure 11.

- Generations: 50
- Individuals: 50
- Best so far: 1336
- Fitness: 0.00000
- Depth: 5
- Nodes: 9

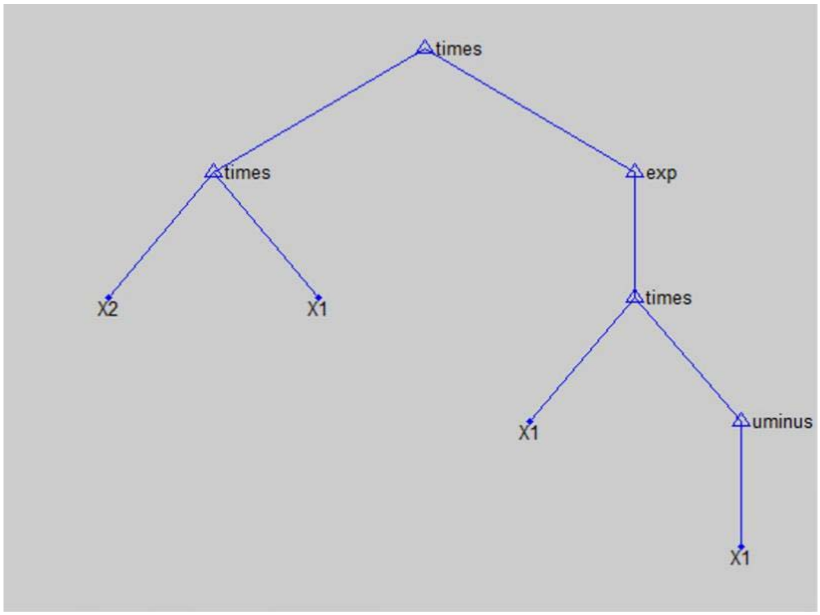

Figure 11: Tree representation of f(x,t), solution of the ODE of example 6.

Figure 12 contains the graph of solution f(x, t) in function (19).

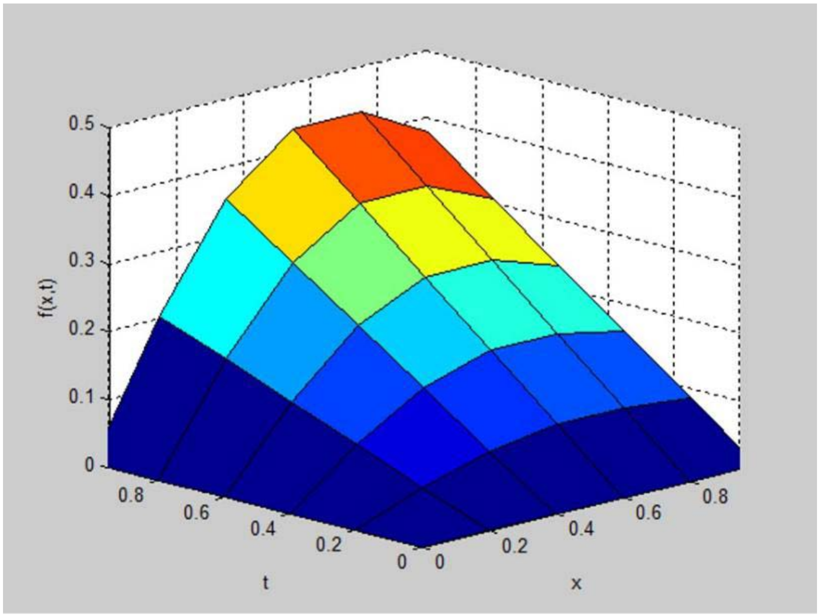

Figure 12: Graph of function f(x,t), solution of the ODE of example 6.

IV. Discussions and Conclusions

This study aimed at investigating the potential of evolutionary algorithms, developed with the combination of GP and AD, to obtain symbolic solutions for problems of differential equations. This is not an issue solely of academic interest. It has great practical relevance, as the ODEs and PDEs are used to model the dynamics of com-plex problems of engineering and of other areas. Examples of research projects applied in engineering include: construction of electrical circuits; modeling of networks cables dilatation; digital filter of "Butterworth", in the area of signal processing; modeling problems of beams; deflection and vibration; mass balance of a chemical reactor; deter-mination of the rate of disintegration or decay of a radioactive element, carbon-14 type; automotive suspensions planning, defining the mechanical damping system; projects of planning and construction of aircraft, turbines, engines and navigation; and industrial projects in general. In addition to engineering, differential equation models are also present and applied in economics, finance, statistics and physics.

With this purpose, GPAD algorithms were developed and a methodology of solu-tion was conceived. Although, in section 3, only six examples have been submitted, we believe this quantity is sufficient to illustrate the results of the applications made. More than twenty problems of ODEs and PDEs, of different types and degrees of dif-ficulty, were used to test the quality and effectiveness of the new methodology. The exact solution was found for all equations tested, with the exception of the EDP for the Schrödinger equation of the helium atom. However, in helium's case, an approxi-mate solution with low-level error was found. The difficulty to solve the equation was expected due to its high complexity.

In order to compare our results with the results of known software, with the ex-ception of the Schrödinger equation, the remaining equations were also tested on the DSolve of Mathematica and in ODE-PDE-Solver Functions of MatLab. The results showed that only the linear ODEs could be solved symbolically. The nonlinear ODEs and PDEs tested, including three PDEs presented in section 3, have not been resolved in symbolic form for these software. Therefore, the results of these tests are very impor-tant, because they show the difficulty of obtaining symbolic solutions of differential equations through computational methods and they confirm the advantages of the GPAD algorithms.

When compared, our study possesses the following similarities with works [2–4]: the examples of ODEs and PDEs used are, in small part, similar to those presented in [6]. This occurred purposefully, because we use three examples of this work as a basis for comparison for evaluating computational efficiency of algorithms developed. However, the similarities are only in the functional form of differential equations, as it was necessary to solve exercises identical to be able to compare effectively the results. These comparisons were of fundamental importance, since they allowed us to pursue better results and develop algorithms that are more efficient.

The modeling and development of the algorithms were overall different, since we used traditional GP method in the form of a tree, while they used the grammatical evolution method. The similarities with articles [2,3] are directly related to the fact that both works intend to solve ODEs and PDEs using the same technique of GP in tree form. However, the examples developed are very different, which leads to different computational modelling.

Based on the results obtained, with exact symbolic solutions for almost all of the problems addressed, we believe that the objective of the study has been achieved and that the GPAD methodology proposed is a new contribution to the literature of evolutionary algorithms and, especially, an effective alternative to assist in the resolution of complex problems of ODEs and PDEs.

Acknowledgement(s)

We would like to thank the Applied Computational Intelligence Laboratory (ICA) at PUC-Rio University for the kindness and collaboration provided whenever requested.

References Références Referencias

1. David CW. A Review of Helium Hamiltonians. Connecticut (CT): Chemistry Education Materials. Connecticut (CT): volume78, pages 1-13. 2009.
2. Imae J., Kikuchi Y., Ohtsuki N. et al. "Design of nonlinear control systems by means of differential genetic programming". In Decision and Control, 2004. CDC. 43rd IEEE Conference on, volume 3, pages 2734– 2739. IEEE, 2004.
3. Luke S. and Panait L. "Lexicographic parsimony pressure", In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, pages 829–836. Morgan Kaufmann Publishers Inc., 2002.
4. Silva S. Gplab a genetic programming toolbox for MatLab, 2009. http://gplab.sourceforge.net/.
5. Makarov D.E. and Metiu H. "Using Genetic Programming to Solve the Schrödinger Equation", The Journal of Physical Chemistry A, vol. 104, no. 37, pp. 8540–8545, 2000.
6. Tsoulos I.G. and Lagari I.E. "Solving Differential Equations with Genetic Programming", Genetic Programming and Evolvable Machines, vol. 7, no. 1, pp. 33 – 54, 2006.
7. Lobão W.J.A., Dias D.M., Pacheco M.A.C. and Pacheco A.C.A. Solving stochastic differ-ential equations through genetic programming and automatic differentiation. Engineering Applications of Artificial Intelligence, v. 68, p. 110-120, 2018.
8. Fink M. Automatic differentiation for MatLab, 2007. http://www.mathworks.com/MatLabcentral/fileexchange/%2015235-automatic-differentiation-for-MatLab.
9. Koza J.R. "Genetic programming: on the programming of computers by means of natural selection", volume 1. MIT press, 1992.
10. Rall L.B. Automatic differentiation: Techniques and applications. Springer, 1981.
11. Griewank A. and Walther A. "Evaluating derivatives: principles and techniques of algo-rithmic differentiation", SIAM, 2008.