Genetic Programming for the Reconstruction of Delay Differential Equations in Economics

Teresa Tonelli
teresa.tonelli@phd.units.it
Dipartimento di Matematica, Informatica e Geoscienze,
Università degli Studi di Trieste
Trieste, Italy

Gloria Pietropolli
gloria.pietropolli@phd.units.it
Dipartimento di Matematica, Informatica e Geoscienze,
Università degli Studi di Trieste
Trieste, Italy

Gabriele Sbaiz
gabriele.sbaiz@deams.units.it
Dipartimento di Scienze Economiche, Aziendali,
Matematiche e Statistiche
Università degli Studi di Trieste
Trieste, Italy

Luca Manzoni
lmanzoni@units.it
Dipartimento di Matematica, Informatica e Geoscienze,
Università degli Studi di Trieste
Trieste, Italy

ABSTRACT

In this paper we explore the reconstruction of a delay differential equation (DDE) model from economics, the Kalecki's business cycle model, with a variant of genetic programming (GP) encoding the structure of DDEs. The results of this preliminary work show that GP can correctly reconstruct the model starting only from approximated data, showing that the investigation of GP for *interpretable* reconstruction of DDEs can be a worthwile research direction.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning approaches*; **Genetic programming**.

KEYWORDS

genetic programming, symbolic regression, differential equations, delay differential equations, Kalecki's business cycle model

ACM Reference Format:
Teresa Tonelli, Gloria Pietropolli, Gabriele Sbaiz, and Luca Manzoni. 2024. Genetic Programming for the Reconstruction of Delay Differential Equations in Economics. In *Gecco '24: The Genetic and Evolutionary Computation Conference, July 14–18, 2024, Melbourne, AU*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3638530.3664164

1 INTRODUCTION

Genetic programming (GP) [14] is a prominent evolutionary computation technique. GP evolves programs, usually represented as trees, to address specific problems given a collection of input and output pairs [7, 22, 26]. Given its success, different versions of GP have been implemented [20, 21] expanding its capability and application range [18, 25, 29]. In this work, in particular, we focus on how to learn interpretable delay differential equations (DDEs) with GP.

Automatic control systems involving feedback control, whose use is more than a century old, require a finite time to process and respond to information. This is mathematically modeled by DDEs, where the evolution of a variable at time t depends on its value at an earlier time $t - \tau$ (with τ representing the delay). The general model for DDEs is thus expressed as $y'(t) = f(t, y(t), y(t - \tau))$ when $t \geq t_0$ and $y(t) = \phi(t)$ when $t \leq t_0$. Here, t_0 denotes the initial time, $\phi(t)$ is the initial condition, and f is a given function, with $y'(t)$ being the derivative of $y(t)$ over time. For an extensive overview of DDEs, readers can refer to [3, 6]. DDEs find applications across various fields, including engineering and economic problems. The economic case we study in this work is Kalecki's business cycle model, a linear first-order DDE with constant coefficients that model macroeconomic dynamics [12]. That model focuses on an endogenous and elementary business cycle mechanism, also incorporating the phenomenon of lag or *gestation period*, which is the amount of time from making an investment decision until the corresponding productive capacity takes place.

In this work, we investigate an approach to dynamic systems learning, a well-established field with a recent interest in data-driven approaches [28]. Specifically, we propose a way to reconstruct DDEs framing them as a symbolic regression problem that we solve using GP. In literature, GP has been applied only to reconstruct ordinary differential equations (ODEs) [8, 11], which are less complex since DDE also depends on past system states. On the other hand, DDE reconstruction has previously been explored through different approaches, such as neural networks [9], leading to good, yet less interpretable, results. One important advantage of using GP instead of deep learning methods relies on the interpretability of the solutions. To accommodate the inherent delays of DDEs, we implement a novel GP algorithm with adjusted tree structures and genetic operators. We then validate our method using the Kalecki's model. Experimental results show that this approach results in a model that closely mirrors the real-world process data.

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0495-6/24/07
https://doi.org/10.1145/3638530.3664164

2 RELATED WORKS

The reconstruction of differential equations from data-driven approaches has seen increasing attention, particularly in the last few years [15]. The ability of GP of describing relationships between data raises the possibility of developing it for the reconstruction of differential equations [16]. Different studies have been proposed: in [5], a GP-based technique solves a system of differential equations through numerical integration, describing the system of equations as a set of trees and optimizing their coefficients through a Genetic Algorithm. In [11], an improvement of this approach is proposed, which uses Runge-Kutta methods of 4^{th} order as a solver and the least mean square error as an optimizer for coefficients. In [4], a grammar-based approach is applied to find the analytical form of a differential equation. Another approach, proposed by [8], adopts symbolic regression to reconstruct the analytical form of ODEs. To the best of our knowledge, no GP-based algorithm has been developed or applied to solve DDEs. Deep learning has also been widely used to reconstruct differential equations [19], initially focusing on ODE, [10, 27] and then expanding to DDEs with recurrent neural networks [30], or with the combination of the adjoint method and neural networks [9]. In this paper we develop a GP method that reconstructs DDEs that takes inspiration from the symbolic regression approach proposed in [8] but that splits the DDEs into the delay and non-delay term, as proposed in [9].

3 DESCRIPTION OF THE ECONOMIC MODEL

In this study, we focus on the DDE represented by the Kalecki's business cycle model [12]. Kalecki's model incorporates a gestation period for productive capacities, modeled by a fixed lag $\tau > 0$. We denote capital stock at time t by $K(t)$ and the investment flow decided at $t - \tau$ by $I(t - \tau)$, leading to the relation $K'(t) = I(t - \tau)$ where $K'(t)$ is the derivative in time of $K(t)$. The model also considers the outstanding stock of orders, which depends on past investment decisions, quantified as: $\int_{t-\tau}^{t} I(s)\, ds$. The corresponding production flow $A(t)$ of investment goods at time t is then given by $A(t) = \frac{1}{\tau}\int_{t-\tau}^{t} I(s)\, ds$. At this point, we postulate a constant aggregate saving propensity $0 < \bar{s} < 1$ such that the total consumption amounts to $(1-\bar{s})Y(t) := cY(t)$. In this way, the aggregate demand of consumption $cY(t)$ and the production flow $A(t)$ equals the total revenue $Y(t)$, i.e. $Y(t) = A(t) + cY(t)$. To close the model, we follow the principle that firms plan new investment projects when demand rises while reducing the investment when the productive capacity increases. That means $I(t) = \lambda(\delta Y(t) - K(t))$ where λ and δ are two positive constant coefficients. After some algebraic manipulations, the system may be recast as a linear first-order DDE with constant coefficients in the variable $K(t)$, as follow: when $t \geq 0$ then $K'(t) = aK(t) - bK(t - \tau)$ and when $t \leq 0$ then $K(t) = 1$, where $a = \frac{\lambda\delta}{\tau(1-c)}$ and $b = \lambda\left(1 + \frac{\delta}{\tau(1-c)}\right)$.

Following the strategy developed in [13], which is based on the differential transform method [1] to solve linear and non-linear differential equations, we get as solution (that we called Keller's approximation):

$$K(t) = 1 - 0.56\, t - 0.4\, t^2 + 0.0533\, t^3 + 0.0266\, t^4 + O(t^5)\,. \quad (1)$$

Constants are taken as in [13], which means $\lambda = 2/5$, $\tau = 1$, $c = 3/4$ and $\delta = 1/2$.

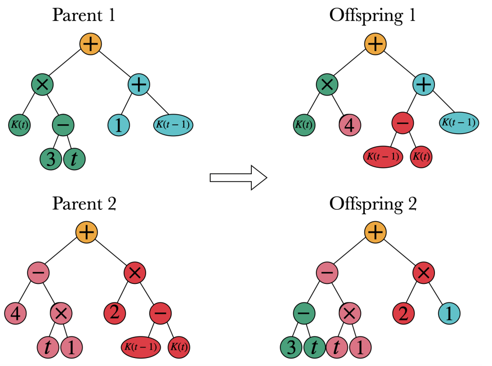

Figure 1: Illustration of the crossover based on the tree structure introduced. The crossover operation is executed distinctly on the left subtree (encoding the non-delay term f_1) and on the right subtree (encoding the delay term f_2).

4 METHODS

This section introduces the GP-based algorithm we develop to solve DDEs. Our goal is to evaluate the effectiveness of the proposed algorithm in accurately reconstructing the DDE structure, also comparing its performance with the analytical approximation detailed in [13]. The algorithm is structured to align both with the architecture of [9] and with DDEs, typically comprising two distinct components: a non-delay term, denoted as f_1, and a delay term f_2, giving $f = f_1 + f_2$.

The proposed algorithm uses a tree structure to mirror the two-part composition of the DDE. Specifically, the GP tree consists of a fixed root node containing the addition operator, linking the subtree corresponding to the non-delay term (f_1) on the left and the subtree corresponding to the delay term (f_2) on the right, reflecting the conventional representation of DDEs as the sum of these two components. Consequently, we define two primitive sets: pset_l for the left subtree and pset_r for the right subtree. The main distinction is the exclusive inclusion of delay elements in the right subtree's primitive set. This novel tree configuration necessitates updates to the GP algorithm, particularly in the crossover and mutation operators, and the subtree pruning function. Crossover operations are now executed distinctly on the left and right subtrees, conducting conventional one-point crossovers within each pair of corresponding subtrees, as illustrated in Figure 1. To ensure the generation of diverse individual by mutation, we employ four different kinds of mutation: uniform, ephemeral, node replacement, and shrink [16]. The standard implementation is maintained, with the distinction that the set of operations that can be used in the mutation process depends on the side of the tree where the mutation is applied: the left primitive set pset_l is used for mutations in the left subtree, and the right set pset_r is used for those in the right subtree. Regarding tree pruning, the usual GP approach evaluates the size of each candidate solution, and if a tree exceeds a predefined maximum size, it cuts the tree at that point. In this version, the nodes at the cut-off depth are replaced with appropriate terminals, selected from the pset_l or the pset_r, based on whether the leaf node being replaced is in the left or right subtree.

5 EXPERIMENTAL SETTINGS

This section outlines the hyperparameters used for the proposed method, ensuring that our experiments can be fully replicated. The source code is available at https://github.com/TeresaTonelli/Gecco_24_workshop/. Data are generated from the Keller's approximation provided in [13]. The population of solutions is composed of 500 individuals with the GP algorithm running for 1000 generations. Both pset$_l$ and the pset$_r$ are built from a set of primitives composed by the three arithmetic operations $\{+, -, *\}$, while the terminals set contain in both cases the time t, the solution $K(t)$ and, exclusively on the pset$_r$, the delay $K(t-\tau)$. Additionally, ephemeral random constants are generated with a uniform probability within the range $[-2.0, 2.0]$ at 0.1 intervals. Each individual is initialized fixing the root and sampling other nodes from the corresponding pset. Crossover operations occur with a probability of $p_c = 0.8$ and all mutations have an equal chance of $p_m = 0.2$. Fitness is measured by the Root Mean Square Error (RMSE) calculated over $N = 101$ grid points which discretize the time t, comparing the derivative approximation by Keller $K'(t_i)$ with the output of the GP individual $f(t_i, K(t_i), K(t_i - \tau))$. Notice that there is no subdivision between a train and test set since the objective is to recover the correct form of the Kaleki's model starting from the data generated from its approximation. Given that the approximation proposed by Keller has an error of $O(t^5)$, it serves as a good approximation for $t \in [0,1]$. Nevertheless, its accuracy decreases for $t > 1$. Therefore, we set the analysis time interval $T = 1$, with time step $\Delta t = 0.01$, using the approximated formula to compute $K(t)$ and its derivative $K'(t)$. The parameters for the model correspond to the ones provided by Keller and described in Section 3; with the delay τ set to 1.

Since GP is inherently stochastic, we conduct 30 runs of the algorithm. To evaluate the change in fitness with the number of generations, we perform a Mann-Whitney U test [17] across all pairs of series of the best individuals at 100, 200, 500, and 1000 generations under the alternative hypothesis that the distribution of the first series of samples produces a fitness that is stochastically greater than the ones in the second series of samples. As a threshold for statistical significance, we select $\alpha = 0.05$.

6 EXPERIMENTAL RESULTS

This section presents the results from applying the GP algorithm to reconstruct Kalecki's model. Figure 2 illustrates boxplots, based on 30 runs, reporting the fitness of the best individuals across generations.

The difference in the results is statistically significant when comparing generation 100 with generations 500 (p-value of 0.0016) and 1000 (p-value of 0.0020). The decreasing trend in fitness confirms the effectiveness of the GP algorithm in navigating the solution space and achieving high accuracy. The fitness median stabilizes around 0.007, which is close to the fitness value of the derivative of Keller's approximation, which is around 0.0035. Moreover, towards the later iterations, some individuals even surpass the Keller approximation, as indicated by the lower ends of the boxplots. Figure 3 compares the GP-generated individuals with the original Kalecki's model and the Keller's approximation. In this comparison, the red curve represents the derivative of Keller's approximation:

$$K'(t) = -0.56 - 0.8t + 0.1599t^2 + 0.1064t^3 + O(t^4) \tag{2}$$

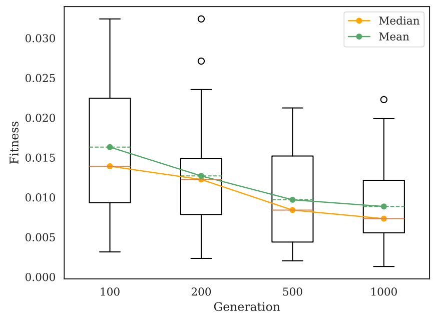

Figure 2: Boxplots (over 30 runs) of the fitness of the best individuals across generations. The orange line represents the trend of the median, while the green one represents the trend of the mean.

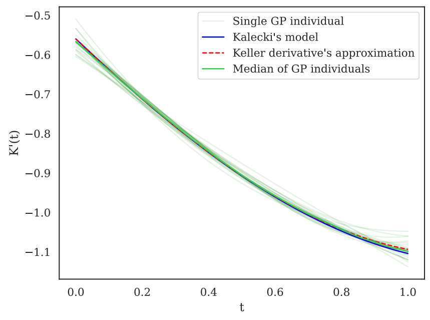

Figure 3: Comparison between Kalecki's model (blue line), Keller's approximation (red line), GP individuals (light green) and their median (darker green).

while the blue represents the Kalecki's model:

$$K'(t) = 0.8\,K(t) - 1.2\,K(t-1)\,. \tag{3}$$

The 30 equations generated by the GP algorithm are plotted in light green, with their median represented in darker green. These results suggest that the proposed GP algorithm can successfully reconstruct DDEs, providing solutions with a behavior close to Kalecki's model. Moreover, the median offers a sensibly better match to the DDE for towards the end of the time range $t \in [0.9, 1]$ compared to Keller's approximation, which becomes less accurate due to the presence of the term $O(t^5)$. The absence of this term in GP solutions provides a closer fit to Kalecki's model in this interval. On the other hand, the early-time accuracy of the GP individuals is slightly lower, due to the absence of initial conditions in the GP algorithm. Kalecki's model presents a set of initial conditions which are taken into account also by the Keller approximation, consequently, our algorithm produces less precise predictions in the initial moments.

To investigate the interpretability of the solutions generated by the GP algorithm we report a (representative) selection of the best individuals at generations 500 and 1 000, noting the frequency of each solution across different runs. At generation 500, solutions include:

$0.8K(t) - 1.2K(t-1)$	freq.: 3 (Kalecki's model)
$2K(t) - 2K(t-1) + t - 0.3$	freq.: 2
$0.7K(t) - K(t-1) - 0.16$	freq.: 1

while, for generation 1000 we have:

$0.8K(t) - 1.2K(t-1)$	freq.: 1 (Kalecki's model)
$-0.4K(t) - K(t-1) - t + 1.0$	freq.: 1
$2K(t) - 2.0K(t-1) + t - 0.3$	freq.: 1

At generation 500, the GP algorithm effectively captures Kalecki's model, with multiple runs producing the exact equation or closely related variations, specifically in a form like $\alpha K(t) - \beta K(t-1) + \gamma$. This confirms the effectiveness of the method in identifying the fundamental structure of the model. On the other hand, for generation 1 000 solutions appear more complex, with just one run matching the exact Kalecki's model. The difference between generation 500 and 1 000 can indicate that GP is overfitting. In fact, this complexity increase could be attributed to the algorithm learning from data generated from the approximation of the real equation, leading to the generation of more elaborate equations as it aims to accommodate the error of the approximation.

7 CONCLUSION

In this paper, we introduce a GP algorithm specifically designed for reconstructing DDEs from data. We propose an algorithm that separates DDEs into two distinct components: one for the delay term and another for the non-delay term. This approach necessitates a specialized architecture for GP trees and genetic operators, which focus on the separate manipulation of each subtree. We validate our method on the Kalecki model and experimental results confirm its ability to deduce underlying equations, yielding some solutions that exactly match the model and others that closely approximate its structure. Moreover, our algorithm can sometimes improve the Keller approximation, especially within the interval $[0.9, 1]$.

This work represents a first attempt to learn DDEs starting from experimental data using GP. Considering the promising results from this initial investigation, this work paves the way for multiple possible future developments. The first possibility relies on using an advanced GP version with adjustable parameters [24], which could be particularly useful when solutions share a common structure but differ in coefficients. To improve the interpretability of the results, we also plan to apply grammatical evolution [23], to fix the structure the solution has to satisfy. An important next step is also to understand when to stop the evolutionary process to avoid learning the approximation errors or the noise in the data, thus obtaining the correct DDEs underlying a dynamical process. Moreover, we plan to extend the analysis to other models like the population dynamics system [2] and structured population DDEs [31].

REFERENCES

[1] Arikoglu, A., and Ozkol, I. Solution of differential–difference equations by using differential transform method. *Applied Mathematics and Computation 181*, 1 (2006), 153–162.

[2] Balachandran, B., Kalmár-Nagy, T., and Gilsinn, D. E. *Delay differential equations.* Springer, 2009.

[3] Bellman, R., and Cooke, K. L. *Differential-difference equations.* Academic Press, 1963.

[4] Bernardino, H. S., and Barbosa, H. J. Inferring systems of ordinary differential equations via grammar-based immune programming. In *International Conference on Artificial Immune Systems* (2011), Springer, pp. 198–211.

[5] Cao, H., Kang, L., Chen, Y., and Yu, J. Evolutionary modeling of systems of ordinary differential equations with genetic programming. *Genetic Programming and Evolvable Machines 1* (2000), 309–337.

[6] Driver, R. D. *Ordinary and delay differential equations*, vol. 20. Springer Science & Business Media, 2012.

[7] Espejo, P. G., Ventura, S., and Herrera, F. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40*, 2 (2009), 121–144.

[8] Gaucel, S., Keijzer, M., Lutton, E., and Tonda, A. Learning dynamical systems using standard symbolic regression. In *Genetic Programming: 17th European Conference, EuroGP 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers 17* (2014), Springer, pp. 25–36.

[9] Gupta, A., and Lermusiaux, P. F. Generalized neural closure models with interpretability. *Scientific Reports 13*, 1 (2023), 10634.

[10] He, S., Reif, K., and Unbehauen, R. Multilayer neural networks for solving a class of partial differential equations. *Neural networks 13*, 3 (2000), 385–396.

[11] Iba, H. Inference of differential equation models by genetic programming. *Information Sciences 178*, 23 (2008), 4453–4468.

[12] Kalecki, M. A macrodynamic theory of business cycles. *Econometrica, Journal of the Econometric Society* (1935), 327–344.

[13] Keller, A. A. Generalized delay differential equations to economic dynamics and control. *American-Math 10* (2010), 278–286.

[14] Koza, J. R. Genetic programming as a means for programming computers by natural selection. *Statistics and computing 4*, 2 (1994), 87–112.

[15] Kutz, J. N. *Data-driven modeling & scientific computation: methods for complex systems & big data.* OUP Oxford, 2013.

[16] Langdon, W. B., and Poli, R. *Foundations of genetic programming.* Springer Science & Business Media, 2013.

[17] Mann, H. B., and Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* (1947), 50–60.

[18] Marchetti, F., Pietropolli, G., Verdù, F. J. C., Castelli, M., and Minisci, E. Automatic design of interpretable control laws through parametrized genetic programming with adjoint state method gradient evaluation. *Applied Soft Computing* (2024), 111654.

[19] Michoski, C., Milosavljević, M., Oliver, T., and Hatch, D. R. Solving differential equations using deep neural networks. *Neurocomputing 399* (2020), 193–212.

[20] Miller, J., and Turner, A. Cartesian genetic programming. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation* (2015), pp. 179–198.

[21] Moraglio, A., Krawiec, K., and Johnson, C. G. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature-PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I 12* (2012), Springer, pp. 21–31.

[22] Nadizar, G., Medvet, E., and Wilson, D. G. Naturally interpretable control policies via graph-based genetic programming. In *European Conference on Genetic Programming (Part of EvoStar)* (2024), Springer, pp. 73–89.

[23] O'Neill, M., and Ryan, C. Grammatical evolution. *IEEE Transactions on Evolutionary Computation 5*, 4 (2001), 349–358.

[24] Pietropolli, G., Camerota Verdù, F. J., Manzoni, L., and Castelli, M. Parametrizing gp trees for better symbolic regression performance through gradient descent. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (2023), pp. 619–622.

[25] Pietropolli, G., Manzoni, L., Paoletti, A., and Castelli, M. Combining geometric semantic gp with gradient-descent optimization. In *European Conference on Genetic Programming (Part of EvoStar)* (2022), Springer, pp. 19–33.

[26] Pietropolli, G., Menara, G., Castelli, M., et al. A genetic programming based heuristic to simplify rugged landscapes exploration. *Emerging Science Journal 7*, 4 (2023), 1037–1051.

[27] Ramuhalli, P., Udpa, L., and Udpa, S. S. Finite-element neural networks for solving differential equations. *IEEE transactions on neural networks 16*, 6 (2005), 1381–1392.

[28] Rudy, S. H., Brunton, S. L., Proctor, J. L., and Kutz, J. N. Data-driven discovery of partial differential equations. *Science advances 3*, 4 (2017), e1602614.

[29] Vasicek, Z. Cartesian gp in optimization of combinational circuits with hundreds of inputs and thousands of gates. In *Genetic Programming: 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings 18* (2015), Springer, pp. 139–150.

[30] Wang, Q., Ripamonti, N., and Hesthaven, J. S. Recurrent neural network closure of parametric pod-galerkin reduced-order models based on the mori-zwanzig formalism. *Journal of Computational Physics 410* (2020), 109402.

[31] Wu, J. *Theory and applications of partial functional differential equations*, vol. 119. Springer Science & Business Media, 2012.