The Solution of Ordinary Differential Equations using Genetic Programming with Constant Tuning

Peter Rockett

School of Electrical and Electronic Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK
p.rockett@sheffield.ac.uk

Abstract. In this paper we report the solution of a benchmark set of ordinary differential equations (ODEs) using genetic programming (GP) within a collocation framework using tuning of the embedded tree constants. We report statistical comparison with a baseline GP approach without constant tuning that indicates that parameter tuning produces statistically superior results. We obtain highly accurate solutions for almost all the benchmark ODEs, but identify a hitherto unreported issue with GP finding trivial solutions. The characteristics of the individual ODEs appear to dictate whether or not solution is problematic.

Keywords: genetic programming, ordinary differential equations, collocation method

1 Introduction

Differential equations (DEs), both ordinary differential equations (ODEs) in one variable and partial differential equations (PDEs) in multiple variables, are central to much of science, engineering and other fields. Unfortunately, many DEs of practical interest have no known solution in terms of elementary functions so numerical approaches via either finite differences or finite elements has been extensively studied. Numerical methods, however, only provide a single solution instance for a given set of initial conditions and at a finite, pre-determined set of mesh points. Other methods aimed at finding analytical, approximate solutions have thus received attention.

Following the seminal work on solving DEs using genetic programming (GP) by Tsoulos and Lagaris [12], the basic DE problem can be described by:

$$g(x, y, y^{(1)}, y^{(2)}, \ldots y^{(n)}) = 0 \quad x \in [a, b] \tag{1}$$

$$\text{subject to} \quad h_j(x, y, y^{(1)}, y^{(2)}, \ldots y^{(n-1)})|_{x=t_j} = 0 \quad \forall j \in [1, p] \tag{2}$$

where $y^{(i)}$ is the i-th order derivative of the unknown solution y with respect to variable x. Eqn (1) is subject to the p constraints described in (2) where

$t_j \in [a, b]$, and the set of h_j functions comprise the problem-specific initial or boundary values. In this initial report, and without loss of generality, we limit our attention to ordinary, 1^{st}- and 2^{nd}-order differential equations (ODEs). The task is to approximate y sufficiently accurately.

The method of *collocation* posits a possible *ansatz* solution ϕ and minimizes a functional of the form:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N_c} \phi(\boldsymbol{\theta})_i^2 \tag{3}$$

where N_c is the number of *collocation points* (or *knots*) in the domain over which the equation is being solved. Classically, easily-differentiable ansatz functions, such as polynomials, have been used and (3) minimized subject to the initial/boundary conditions in (2).

Recently, there has been much interest in the machine learning literature in solving DEs using deep neural networks (DNNs) as ansatz solutions due to their flexibility and representational power. DNNs, however, have the significant disadvantages of requiring large computing power, especially in training. For this, and other reasons, solving DEs using much more compact genetic programming (GP) models is of great interest.

Previous work on DEs using genetic programming (GP) has been fairly limited. The seminal work of Tsoulos and Lagaris [12] minimized a loss functional:

$$\mathcal{L} = \sum_{k=1}^{N_c} g^2(x_k) + \lambda P \tag{4}$$

where P is the sum of the squares of the p constraint violations specified in (2), and λ is a user-selected non-negative weight; the x_k's are a set of N_c equally-spaced collocation points $x_k \in [a, b]$. We have followed [12] and used λ-values of unity in this work since there appears no principled way of optimizing the λ other than grid search. Seaton et al. [11] have used a similar approach to [12] but using Cartesian GP.

Lobão et al. [6] have also approximated the solutions to DEs by minimizing a fitness functional comprising the sum of the maximum absolute error (MAE) between the given DE and the corresponding DE formed using AD from a candidate solution, plus the magnitudes of the deviations from the initial conditions; the MAE was evaluated on a pre-determined grid of, typically 50, points. Their formulation of the fitness function is thus similar to [12].

In this paper, we explore the research question of whether tuning the constant values in the GP trees during evolution of solutions to a range of benchmark ordinary differential equations (ODEs) that have been studied before in the literature. Previous work on symbolic regression problems [9] showed that tuning produces statistically significantly better results as well as smaller trees. Unlike previous contributions to the GP solution of ODEs, here we also report statistical comparisons of GP with and without constant tuning; GP *without* tuning is the baseline method previously employed by Tsoulos and Lagaris [12] and others.

Further, we also explore the subsidiary research question of whether a restricted function set is able to synthesize ODE solutions. As with symbolic regression, we find that constant tuning produces superior results although we identify an issue with the GP collocation method generating trivial (*i.e.* $y = 0$) solutions that, as far as we aware, has not previously been reported.

2 Methodology

2.1 Benchmark Ordinary Differential Equations

We used the set of benchmark ordinary differential equations (ODEs) in Table 1 and previously studied in [12, 11]. For what follows, we note that ODE-4, ODE-6, ODE-7 and ODE-8 all have the possible trivial solution of $y = 0$ notwithstanding the initial conditions. Like [12, 11], we have used 50 collocation points in the present work.

Table 1: Benchmark ordinary differential equations used in this work, after [11]. The second column shows the initial conditions.

ID	ODE	Domain	Initial conditions
ODE-1	y' $= \frac{2x-y}{x}$	$\forall x \in [0.1, 1]$	$y(0.1) = 20.1$
ODE-2	$y' = \frac{1-y\cos(x)}{\sin(x)}$	$\forall x \in [0.1, 1]$	$y(0.1) = \frac{2.1}{\sin(0.1)}$
ODE-3	$y' = \frac{-y}{5} + e^{\frac{-x}{5}}\cos(x)$	$\forall x \in [0, 1]$	$y(0) = 0$
ODE-4	$y' = -y\cos(x)$	$\forall x \in [0.1, 1]$	$y(0) = 1$
ODE-5	$y' = x + \frac{2y}{x}$	$\forall x \in [0.1, 1]$	$y(1) = 10$
ODE-6	$y' = -y^2$	$\forall x \in [0, 1]$	$y(1) = 0.5$
ODE-7	$y'' = -100y$	$\forall x \in [0, 1]$	$y(0) = 0,\ y'(0) = 10$
ODE-8	$y'' = 6y' - 9y$	$\forall x \in [0, 1]$	$y(0) = 0,\ y'(0) = 2$
ODE-9	$y'' = -\frac{y'}{5} - y - \frac{1}{5}e^{-\frac{x}{5}}\cos(x)$	$\forall x \in [0, 2]$	$y(0) = 0,\ y'(0) = 1$
ODE-10	$y'' = 4y' - 4y + e^x$	$\forall x \in [0, 1]$	$y(0) = 3,\ y'(0) = 6$

2.2 Genetic Programming Environment

The evolutionary algorithm used in this work is identical to that we have employed previously [9]; the algorithm's key parameters are shown in Table 2. We have used a steady-state, as opposed to generational, evolutionary algorithm since our experience is that this give superior results. The bi-objective fitness vector comprised i) the value of the loss functional, and ii) the tree's node count as an approximate measure of model complexity. Parents were selected for breeding by sorting the population by Pareto rank, and mapping an individual's rank to a scalar with a linear function such that the best-ranked individuals received the largest value and the worst ranked received zero. Selection of the parents

Table 2: Evolutionary algorithm parameters used in this work

Parameter	Value
Evolutionary strategy	Steady-state
Population size	100
Initialization method	Uniformly-random tree node counts $\in [16 \dots 128]$
Function set	Unary minus, $+$, $-$, $\times$, Analytic Quotient (AQ) [7]
Terminal set	Input variable and mutable constants
Initial mutable constants	Uniformly selected $\in \{0.1, 0.2, \dots, 0.8, 0.9\}$
Objectives to minimize	i) Loss functional and ii) Tree node count
SLSQP convergence	50 iterations or objective $\leq 10^{-4}$
Boundary/initial value tolerance	10^{-6}
Selection	Pareto ranking – see [3]
No. of children	Two children produced per breeding operation
Crossover	Point crossover; probability Pr = 0.9 of selecting an internal node
Mutation	Subtree mutation [8, p.16] (full trees of depth = 4)
Mutation probability	1.0
Total number of generated children	10,000

was then performed stochastically biased by these scalar values—see [3, p.32] for full details.

Each execution of the GP program generated 10,000 offspring, and the tree with the smallest value of loss functional taken as the best-of-run individual regardless of tree size. We generated a set of 30 different initial populations and ran each GP setup once for each population. We have used the same constant tuning approach as in previous work [9]. The required derivatives were calculated using automatic differentiation, and we used Sequential Least-Squares Quadratic Programming (SLSQP) as the non-linear optimizer[1].

Restricting the function set to: unary minus, $+$, $-$, $\times$ and the AQ operator [7] was deliberate, and designed to address an explicit research question. The evolutionary search may have been made easier by including functions such as sin and cos, but we wish to avoid the criticism that can be leveled at [12], for example, of embedding the exact solutions in the problem formulation. In practice, large numbers of DEs have no known solution in terms of elementary functions so a solution needs to be synthesized: selecting a function set to efficiently solve a set of *benchmark* problems with known solutions lacks generality; the Weierstrass approximation theorem [4], on the other hand, states that any continuous function can be uniformly approximated by a polynomial. We have thus deliberately limited the function set to explore the ability of GP to synthesize solutions that may not contain elementary functions.

[1] We have used the implementation of SLSQP from version 2.7.1 of the NLOpt library (https://nlopt.readthedocs.io/en/latest/).

2.3 ODE Problem Formulation

Rather than using automatic differentiation to formulate the ODE problem, it is more convenient to directly generate trees that evaluate the derivative of a tree function using an exact tree transformation method [10] based on the chain rule of calculus, and therefore accurate to rounding error. Given a would-be solution $u(x)$, say, we generate an independent tree that evaluates the derivative function $u_x(x)$, where u_x is the derivative w.r.t x. If required, we generate u_{xx}, the second derivative w.r.t x by applying the tree differentiating transformation to u_x. Thus we can straightforwardly evaluate the $g(u_{xx}, u_x, u, x)$ term in (4) in order to calculate the fitness of the potential solution u.

2.4 Statistical Testing

To compare the results of the GP solution of ODEs with/without constant tuning, we have adopted two statistical approaches. We have used a null hypotheses test, as is near-universal in the GP literature, although here we employ a Bayesian test procedure. The shortcomings of conventional frequentist null hypothesis statistical tests (NHSTs) have been repeatedly highlighted [5]; in addition, frequentist NHSTs are commonly misinterpreted, and subject to 'black-and-white' thinking [5]. Further, since NHSTs make a decision on significance (usually at the $p = 0.05$ level), multiple comparison corrections to the p-value are needed to constrain the overall error rate. As an alternative, we have employed a Bayesian Wilcoxon signed-rank test [1] to estimate the posterior probability density $p(H|\mathcal{D})$ of the null hypothesis H given the observed data $\mathcal{D}$[2]. As is conventional, we have paired the results by initial population. Since we are not making decisions about statistical significance, Bayesian analysis does not require any multiple comparison corrections. Further, unlike frequentist NHSTs, Bayesian tests provide evidence to accept the null hypothesis rather than the much weaker statement that the null hypothesis cannot be rejected.

Notwithstanding, the finding of statistical significance from a hypothesis test does not necessarily imply any meaningful difference between the two groups [5]. Therefore, we have examined two *effect sizes*: First, a key criterion that may be used by an analyst to select between competing methods is the upper bound on the magnitude of the relative errors: the smaller the most extreme error, the better. For the 30 repetitions in each group—with and without parameter tuning—we have determined the magnitudes of the largest relative errors for each best-of-run result, and calculated the median difference between the paired results. To gauge effect size, we have estimated the 5%/95% confidence interval (CI) on this point measure using bias-corrected and accelerated bootstrap resampling [2][3]. If the null hypothesis value of zero difference falls within the

[2] For convenience, we report below the posterior probability that GP with tuning performs better than without.

[3] We have used the bootstrap implementation from SciPy version 1.15.3 (https://docs.scipy.org).

CI then the two methods can be interpreted as not meaningfully different; the converse is obviously true.

The second set of effect sizes we have examined relates to how well the initial condition(s) of each ODE have been satisfied. We have calculated the absolute errors on each of the initial conditions for the two groups of 30 paired repetitions, determined their median paired differences, and again estimated the confidence intervals by bootstrap resampling. As before, our motivation is that, all other things being equal, an analyst would prefer methods that satisfied the initial conditions more accurately.

While the statistical testing described above provides information on the relative performance of the with/without constant tuning methods 'on average', we also include a selection of the 'best' results we have obtained for a range of benchmark ODEs. These were selected on the basis of yielding the smallest *training* errors rather than the best test performance since this is more realistic in the practical situation where the true ODE solution is unknown.

3 Results

3.1 Statistical Comparisons

The results of statistical comparisons for the first-order ODEs are summarized in Table 3 and the second-order ODEs in Table 4.

The layouts of these two tables are similar: each contains Pr(Test error), the probability of the differences between the paired test errors being <0. Similarly, the second column shows Pr(Initial condition), the probability that the values of the initial conditions achieved by GP-with-tuning are superior to those without constant tuning. For the 2^{nd}-order ODEs (Table 4), the third column shows Pr(Initial derivative), the probability that the values of the initial derivatives are more closely achieved by GP-with-tuning. Values of 1.0 for these three probabilities indicate, with as close to certainty as can be predicted with a statistical test, that the null hypotheses can be rejected, and that GP-with-tuning is superior.

As noted in Section 2.4, null hypothesis testing alone is insufficient so Tables 3 and 4 also show bootstrapped estimates of the confidence intervals for: the median paired differences in test error, the median paired differences in the initial values violations, and, for the case of the 2^{nd} order ODEs, the median paired differences in initial derivative value violations. For each ODE, the confidence interval (CI) results are shown over three lines: the 5% percentile, the notional value of the median, and the 95% percentile. So, for example, the CI for the median relative error for ODE-1 is $[-0.60, -0.45, -0.36]$. Since this interval does not include the value of zero, one can infer that the effect size between GP with and without tuning is significant with the with-tuning variant performing consistently better.

Turning for to Table 3, it appears from the CIs that GP without constant tuning performs particularly poorly on ODE-1 and ODE-2, which is surprising for such simple ODEs. For the other ODEs, however, the effect sizes are of the

order of 10^{-2} and taking the interval of $[-10^{-6}, +10^{-6}]$ as the region-of-practical equivalence (ROPE), almost all the results indicate a significant effect size.

Table 3: Statistical results for the first-order benchmark ODEs. The first column shows the ODE. The second and third columns show the posterior probabilities that the method with constant tuning has a better validation error and attainment of the initial conditions, respectively. The fourth column shows the confidence intervals for the median differences between the maximum absolute errors. The fifth column shows the confidence intervals for the median differences of the absolute errors to which the initial condition were satisfied.

ODE	Pr (Test error)	Pr (Initial condition)	Median error difference CI	Median initial value CI
ODE-1	1.0	1.0	-0.60	-5.24
			-0.45	-4.21
			-0.36	-3.40
ODE-2	1.0	1.0	-0.51	-5.38
			-0.42	-4.17
			-0.31	-3.52
ODE-3	1.0	4.2×10^{-4}	-1.77×10^{-2}	0.00
			-1.33×10^{-2}	1.08×10^{-15}
			-8.25×10^{-3}	4.39×10^{-13}
ODE-4	0.96	1.0	-1.71×10^{-2}	-1.21×10^{-2}
			-1.00×10^{-2}	-6.27×10^{-3}
			-3.88×10^{-3}	-4.07×10^{-4}
ODE-5	1.0	1.0	-9.12×10^{-2}	-2.51×10^{-2}
			-8.33×10^{-2}	-1.51×10^{-2}
			-6.46×10^{-2}	-1.02×10^{-2}
ODE-6	1.0	1.0	-0.23	-4.79×10^{-2}
			-0.17	-3.41×10^{-2}
			-0.126	-2.54×10^{-2}

The results in Table 3 generally indicate the, often large, superiority of GP-with-tuning although there is one notable inconsistency: the performance of GP-without-tuning on ODE-3 where both the null hypothesis test and the CI suggest no difference between the tuned and no-tuning variants for satisfaction of the initial value although the test error results do show significant advantage for GP-with-tuning. We have examined the distribution of the initial condition violations for the without-tuning variant on ODE-3, and this reveals that all the values are identically zero. The with-tuning variant, on the other hand, had a corresponding distribution in the range of 0.0 to $2,5\times10^{-10}$ hence no statistical difference. We investigated this initial condition result further by replacing value of zero shown in Table 1 with two small values of $\pm10^{-3}$ and re-running the GP experiments.

These small changes in initial value now produce clear superiority of the with-tuning GP as the Pr(Initial value) values are both 1.0; the CIs are now $[-1 \times 10^{-3}, -1 \times 10^{-3}, -1 \times 10^{-3}]$ and $[1 \times 10^{-3}, 1 \times 10^{-3}, 1 \times 10^{-3}]$, respectively. (The CI results here are explained by the initial value violations falling in the scale of $\sim 10^{-10}$ for the with-tuning variant while the values for the without-tuning variant are all $\mp 10^{-3}$, hence the differences are still $\mp 10^{-3}$ after rounding.) We thus conclude that the apparently good performance on the initial value satisfaction for ODE-3 shown in Table 3 is a coincidence. A small variation of the initial value to one that cannot be (fortuitously) generated using the terminal set alone does not appear to be learned without tuning the constants in the trees.

Turning to the 2^{nd}-order ODE results in Table 4, the result for ODE-7 is omitted and will be discussed in greater detail below.

Table 4: Statistical results for the second-order benchmark ODEs. The first column shows the ODE. The second and third columns show the posterior probabilities that the method with constant tuning has a better validation error and attainment of the initial conditions, respectively. The fourth column shows the confidence intervals for the median differences between the maximum absolute relative errors. The fifth and sixth column shows the confidence intervals for the median differences of the absolute errors to which the initial and derivative condition, respectively, were satisfied.

ODE	Pr (Test error)	Pr (Initial condition)	Pr (Initial derivative)	Median error difference CI	Median initial value CI	Median initial derivative CI
ODE-8_t	1.0	1.0	1.0	-1.02×10^{3}	-1.03	-1.45
				-7.75×10^{2}	-0.776	-0.769
				-5.36×10^{2}	-0.536	-0.487
ODE-9	1.0	0.042	1.0	-4.88×10^{-2}	0.0	-1.85×10^{-2}
				-3.41×10^{-2}	8.75×10^{-15}	-9.24×10^{-3}
				-3.22×10^{-2}	9.65×10^{-11}	-1.74×10^{-3}
ODE-10	1.0	1.0	1.0	-3.67×10^{-1}	-7.20×10^{-1}	-5.17×10^{-1}
				-3.46×10^{-1}	-6.60×10^{-1}	-4.51×10^{-1}
				-3.19×10^{-1}	-5.94×10^{-1}	-3.31×10^{-1}

It was noted in Section 2.1 that ODE-4, ODE-6, ODE-7 and ODE-8 admit trivial (*i.e.* $y = 0$) solutions notwithstanding the initial values. For ODE-7 and ODE-8, both with- and without-tuning GP variants produced solutions that were all trivial. On investigation, all the solutions being generated had the typical form illustrated in Figure 1 where $g(x_k) \approx 0$ is satisfied at all the internal collocation points, and the would-be solution matches the initial condition at the edge of the domain. In essence, GP is finding a solution to the *collocation problem* rather than the ODE. As far as we are aware, this generation of trivial solutions has not previously been mentioned in the literature.

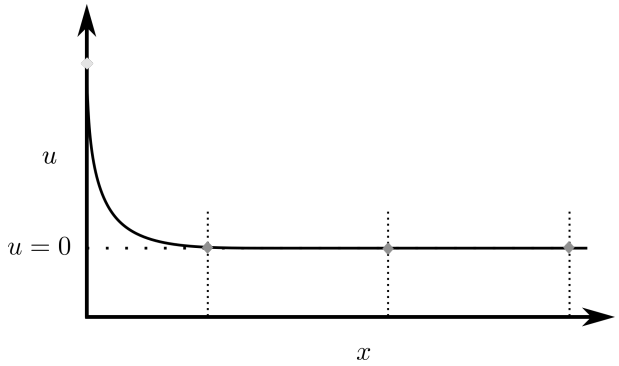


Fig. 1: Illustration of the typical trivial solution. The darker diamonds are collocation points and the lighter diamond the initial value.

To suppress the creation of trivial solutions, we modified the loss function in (4) to include an additional soft penalty constraint:

$$\sum_{k=1}^{N_C}(u_k)^2 - T \geq 0 \tag{5}$$

to 'encourage' the solution away from all zero values by constraining the 'energy' contained in the evolved solution to be $\geq T$, where T is a positive threshold.

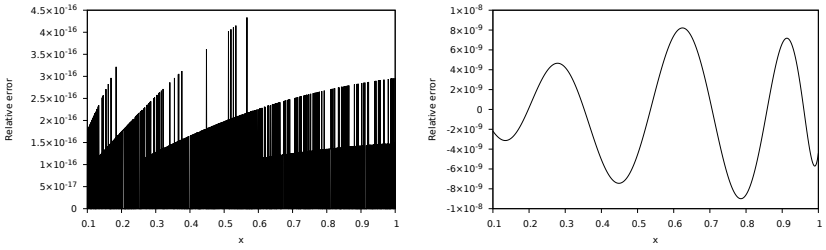
For ODE-8, we used a value for $T = 100$, this being roughly half of the test error generated by the trivial solution. The result is recorded in Table 4 as 'ODE-8_t' ('t' for thresholded). Adding this threshold term allowed both the with-and without-tuning GP variants to produce the desired, non-trivial solutions although the results in Table 4 indicate that again with-tuning is notably superior.

Attempting to add the additional 'energy' term, however, to ODE-7 failed to generate any non-trivial solution regardless of the value of threshold value employed. We return to discussing this ODE in Section 4.

3.2 Best Approximation Results

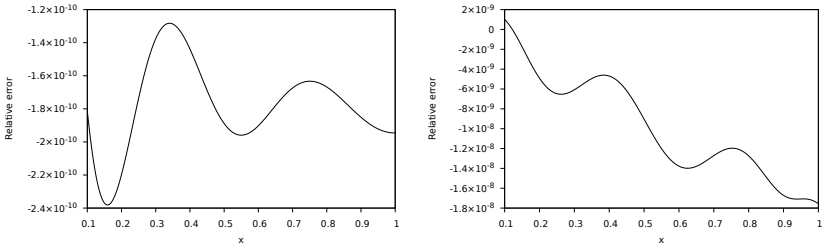
In this section we show a selection the best solutions obtained for a range of ODEs. We reiterate that we have selected the 'best' results on the basis on the smallest training error obtained, not the best test error. We also show plots of *relative* error since these are more useful for an analyst.

Figure 2a for ODE-1 appears surprising at first glance although from examining the ordinate scale and noting that the calculations have been performed with double-precision arithmetic, it is clear that the solution has been found to within rounding error. This is interesting since the GP function set used cannot represent the solution to this ODE exactly; clearly the approximation is very good. The best solution to ODE-6—not shown—also exhibits rounding



(a) ODE-1 (b) ODE-4

Fig. 2: Relative error performance of the best generated solutions to (a) ODE-1 and (b) ODE-4.



(a) ODE-9 (b) ODE-10

Fig. 3: Relative error performance of the best generated solutions to (a) ODE-9 and (b) ODE-10.

error 'noise' although the other 'best' solutions typically exhibit the oscillatory behavior shown in Figures 2b and 3, and common in function approximation. In fact, the only 'best' performance that fell outside the 10^{-6} region of probable equivalence criterion was for ODE-2 where the relative error ranged over $[-1 \times 10^{-5}, 2 \times 10^{-5}]$. Nonetheless, these appear to be highly accurate solutions.

4 Discussion and Future Work

The principal issue arising from this work is the difficulty in generating a satisfactory solution for ODE-7. It should be noted that this ODE was successfully solved in [12, 11] although both these authors included the sin function in their function sets—the analytical solution is $y = \sin(x)$. We therefore conjecture that both these papers reported simple solutions that included only the sin function. As we have noted in Section 2.2, pre-configuring the GP function set to match all possible solutions to a set of benchmark problems lacks generality.

More generally, ODE-4, ODE-6, ODE-7 and ODE-8 all admit trivial solutions although ODE-4 and ODE-6 yielded the desired solution without any modification. ODE-8 required the addition of a minimum 'energy' threshold term to

the soft penalty to steer the solution away from trivial solutions. However, this approach did not work for ODE-7. We therefore infer that the desired solutions to these ODEs have some 'basin of attraction'. For ODE-4/ODE-6, the basin is easy to 'fall into'. For ODE-8 it can be located given a minor stimulus, but for ODE-7 locating the basin appears challenging. Rather than imposing a soft (*i.e.* optional) constraint on the minimum energy contained in the solution, the obvious approach is to impose a *hard* constraint on the evolutionary process.

As a supplementary experiment, we have included the energy constraint for ODE-7 not in the soft penalty term in (4) but as a so-called 'death penalty' in the evolutionary optimization whereby if a solution fails to meet the constraint it is allocated the maximum fitness and thus very likely to be rapidly eliminated from the population. The death penalty is simple to implement, widely used in evolutionary computation (EC), but has the major disadvantage that if none of the members of the initial population meet the constraint(s), the EC degenerates to inefficient random search. We observed that when we tried the death penalty, almost all of the initial populations were infeasible and therefore search performance was likely to be poor. Nonetheless, the death penalty did produce some partially successful results, the relative errors for which are shown in Figure 4 where the error values around zero are on the scale of 10^{-5} but the function has a number of overly-complex 'spike' features. We conjecture that, given more efficient learning, the EC would remove these 'spikes' since a multiobjective GP acts to prefer simpler solutions given the same training error. Clearly an area for future research is to facilitate better imposition of hard constraints than the death penalty method. A corollary of Figure 4 is that given more efficient search to remove the 'spike' features and an appropriate steer into the desired basin of attraction with the imposition of a hard constraint, the limited function set employed in this work does seem able to synthesize arbitrary ODE solutions.

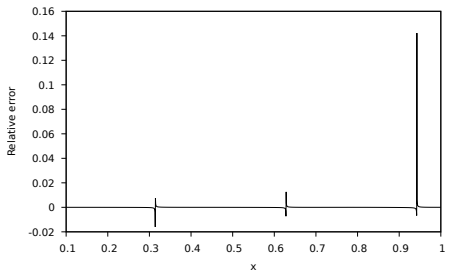


Fig. 4: Typical relative errors for ODE-7 with hard constraints imposed using the 'death penalty'.

The other obvious area of future work is to explore the influence of varying the numbers of collocation points. Here we have used 50 points, the same as previous work, but it is not clear if this is too many or too few (for a given ODE). In addition, extension to partial differential equations is straightforward.

5 Conclusions

In this paper we have applied constant tuning to the solution of ordinary differential equations (ODEs) using genetic programming. In general, and consistent with previous work on symbolic regressions problems [9], tuning of the constants during evolution generally produces statistically superior results. We have identified a particular issue with ODEs that admit of trivial solutions, and observe that there is different behavior across different ODEs, presumably as a consequence of the equation's properties: some seem to require no intervention, some yield the desired, non-trivial solution with a simple modification to the soft penalty, whereas some appear to need the imposition of hard constraints on the form of the solution.

References

1. Benavoli, A., Corani, G., Mangili, F., Zaffalon, M., Ruggeri, F.: A Bayesian Wilcoxon signed-rank test based on the Dirichlet process. In: 31st International Conference on Machine Learning. vol. 32, pp. 1026–1034 (2014)
2. Efron, B.: Better bootstrap confidence intervals. Journal of the American Statistical Association 82(397), 171–185 (1987)
3. Fonseca, C., Fleming, P.J.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: A unified formulation. IEEE Transactions on Systems, Man & Cybernetics - Part A: Systems & Humans 28(1), 26–37 (1998)
4. Jeffreys, H., Jeffreys, B.: Methods of Mathematical Physics. Cambridge University Press, Cambridge, 3rd edn. (1999)
5. Kruschke, J.K., Liddell, T.M.: The Bayesian New Statistics: Hypothesis testing, estimation, meta-analysis, and power analysis from a Bayesian perspective. Psychonomic Bulletin & Review 25(1), 178–206 (2018)
6. Lobão, W.J.A., Dias, D.M., Pacheco, M.A.C.: Genetic programming and automatic differentiation algorithms applied to the solution of ordinary and partial differential equations. In: IEEE Congress on Evolutionary Computation. pp. 5286–5292 (2016)
7. Ni, J., Drieberg, R.H., Rockett, P.I.: The use of an analytic quotient operator in genetic programming. IEEE Transactions on Evolutionary Computation 17(1), 146–152 (2013)
8. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Published via http://lulu.com (2008)
9. Rockett, P.: Constant optimization and feature standardization in multiobjective genetic programming. Genetic Programming and Evolvable Machines 23(1), 37–69 (2021)
10. Rockett, P., Kaszubowski Lopes, Y., Dou, T., Hathway, E.A.: d(Tree)-by-dx: Automatic and exact differentiation of genetic programming trees. In: 14th International Conference on Hybrid Artificial Intelligent Systems (HAIS2019). pp. 133–144 (2019)
11. Seaton, T., Brown, G., Miller, J.F.: Analytic solutions to differential equations under graph-based genetic programming. In: 13th European Conference on Genetic Programming (EuroGP'10). pp. 232–243 (2010)
12. Tsoulos, I.G., Lagaris, I.E.: Solving differential equations with genetic programming. Genetic Programming and Evolvable Machines 7(1), 33–54 (2006)