White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Sheffield.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/238889/

Version: Published Version

Article:
Rockett, P. (2026) Solving ordinary differential equations with genetic programming with hard initial/boundary value constraints. Genetic Programming and Evolvable Machines, 27 (1). 11. ISSN: 1389-2576

https://doi.org/10.1007/s10710-026-09536-x

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:
https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

Genetic Programming and Evolvable Machines (2026) 27:11
https://doi.org/10.1007/s10710-026-09536-x

Solving ordinary differential equations with genetic programming with hard initial/boundary value constraints

Peter Rockett[1]

Received: 24 September 2025 / Revised: 3 March 2026 / Accepted: 5 March 2026
© The Author(s) 2026

Abstract

We report the solution of a benchmark set of ordinary differential equations (ODEs) with genetic programming (GP) within a collocation framework using numerical tuning of the embedded tree constants. Alongside a conventional soft penalty formulation, we also report results from two GP variants that enforce the initial conditions on the ODEs as hard constraints: the first uses the so-called death penalty while the second employs a novel ranking method that orders infeasible individuals using Pareto dominance according to the degree to which they violate the constraints. We investigate the influence of the numbers of collocation points used to solve the problem, and conclude that a few ODEs require more than 10–20 points, otherwise the number of points is not critical. A statistical comparison of the different methods indicates that only a few ODEs display differences, an observation we attribute to the influence of parameter tuning. We obtain highly accurate solutions for all the benchmark ODEs, but identify a problem with certain of the ODEs producing trivial solutions, which we are able to mostly mitigate by introducing an additional constraint on the mean squared amplitude of the evolved solutions. Overall, we infer that the properties of the individual ODEs can impact the solution process.

Keywords Genetic programming · Ordinary differential equations · Collocation methods · Constant tuning

✉ Peter Rockett
p.rockett@sheffield.ac.uk

[1] School of Electrical and Electronic Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK

Published online: 27 March 2026

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; the Duffing equation [1], for example, is a generalization of the linear harmonic oscillator to include a nonlinear restoring term, but does not appear to have any general analytical solution. Consequently, numerical approaches to solve DEs via either finite differences or finite elements have been extensively studied. Numerical methods, however, yield only a single solution instance on 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 [2], the basic ODE 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. Eq. (1) is subject to the p constraints described in Eq. (2) where $t_j \in [a, b]$, and the set of h_j functions comprise the initial or boundary values, depending on the particular problem. In this paper, and without loss of generality, we restrict our attention to the set of benchmark ordinary, first- and second-order differential equations (ODEs) previously studied in [2, 3]. The task is to approximate y sufficiently accurately. Note that there is no training dataset involved in this process.

The method of *collocation* posits a possible solution—or *ansatz*—$\phi(x, \boldsymbol{\theta})$ parameterized in $\boldsymbol{\theta}$, and minimizes a functional of the form:

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{N_c} [\phi(x_k, \boldsymbol{\theta})]^2 \tag{3}$$

where N_c is the number of so-called *collocation points* (or *knots*) in the domain [a, b] over which the equation is being solved. Classically, an easily-differentiable ansatz function, such as a polynomial, is used, and Eq. (3) minimized subject to the initial/boundary conditions in Eq. (2). The challenge, of course, is in identifying a sufficiently flexible ansatz without producing either underfitting or overfitting.

Recently, there has been great interest in the machine learning literature in using deep neural networks (DNNs) as ansatz solutions [4] due to fact they are universal approximators. DNNs, however, have the significant disadvantage of requiring large computing power, especially in training, due to the large number of free parameters (weights). Determining the network architecture—the number of layers/neurons per layer—is also an issue. As a consequence, solving DEs using much more compact

genetic programming (GP) models is of great interest. In this paper, we apply GP to synthesize an ansatz to solve the collocation problem in Eq. (3).

Previous work on solving DEs using genetic programming (GP) has been fairly limited. The seminal work of Tsoulos and Lagaris [2] minimized a loss functional:

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

where P is the sum of the squares of the p constraint violations specified in Eq. (2), and λ is a user-selected non-negative weight; the x_k's are a set of N_c equally-spaced collocation points $x_k \in [a, b]$. Tsoulos and Lagaris evaluated the derivatives in Eq. (1) using automatic differentiation (AD) [5]. Aside from possible dependence on the choice of the N_c sampling points at which the DE is evaluated, it is clear from Eq. (4) that the choice of λ could influence whether the evolved solution favors a small value of the first goodness-of-fit term in Eq. (4) but allows a larger value of the second, constraint-violation term (small λ). Or vice versa (large λ). Tsoulos and Lagaris used values of $\lambda = 100$, but did not justify this choice—presumably, it was arrived at empirically. There is no way of finding an optimal value of λ in this trade-off [6, 7], other than exhaustive grid search, and thus an accurate balance between simultaneously matching the initial/boundary values and the accuracy of the first DE error term in Eq. (4) is unclear. Seaton et al. [3] have employed a similar approach as [2] except using Cartesian GP; they explicitly noted the "critical" role of the weighting constant λ on the solutions evolved.

Lobão et al. [8] 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, and the formulation of the fitness function was similar to [2]. As with the work cited above, there is a potential trade-off between accuracy of the DE solution and meeting the initial conditions. In addition, the dependency on the number of points used to search for the MAE is unclear.

Oh et al. [9] have presented a GP technique that weights $n \in [2, 5]$ (numerically-obtained) training data with $m = 11$ point evaluations of the differential equation (as in [2]), and minimizing the mean-squared error of the resulting combined measures while also optimizing the constants of the GP tree.

In related work, we have previously demonstrated [10] that including constant tuning in GP produces statistically superior ODE results, a finding that mirrors previous results for GP regression problems [11].

In this paper, we extend the research in [10] that only considered constant tuning[1] with a loss functional like Eq. (4). Since the initial/boundary value ODE constraints are included as (soft) penalties, in what follows we refer to the GP framework that

[1] The term *parameter* tuning would be used in the mainstream statistical learning literature as distinct from *hyperparameter* tuning, which would refer to adjusting the parameters associated with the learning

minimizes Eq. (4) as the *penalty method* (**pm**). Ideally, the boundary/initial values (BIVs) should be matched *exactly* with no scope for any trade-off against the quality of the general ODE solution that might occur with Eq. (4). Thus it is desirable to formulate the problem using *hard constraints* [12], which must be met, as opposed to the soft, optional constraint approach exemplified by Eq. (4). This comment applies equally to DNNs, for example, the work of Raissi et al. [4]. In fact, including hard constraints in DNNs has been explored by Lu et al. [13] using an augmented Lagrangian solver, and by Márquez-Neila et al. [14]. In this paper, we investigate two methods of imposing initial conditions as hard constraints in GP.

The present work is builds upon on three previous contributions: Firstly, the use of an analytic quotient (AQ) operator [15] instead of (protected) division. This is an important feature of our formulation since the use of a function set containing the AQ operator [15] guarantees that all derivatives of any arbitrary tree exist, i.e. any tree is $\in \mathcal{C}^{\infty}$. The same is not true of trees constructed with the normal division operator, whether protected or otherwise. It is noteworthy that Oh et al. [9], who employed an ordinary division operator, encountered problems with the non-differentiability of some evolved trees.

Secondly, the collocation approach described above requires the minimization of the loss of the approximating model with respect to its parameters, here the values of the constants embedded in a given tree. Recent work on constant optimization in GP trees has been reviewed in [11]. Although the well-known Levenberg-Marquardt algorithm is a common choice for nonlinear least-squares optimization, it is known to lack robustness due to potential rank-deficiency problems [16]. Like [11], in the present work, we have used a sequential quadratic programming (SQP) based method[2] which is known to be more robust albeit somewhat slower. It is also able to optimize quite general objective functions, not just the sum of squared errors. More importantly, for the present application, SQP methods allow the incorporation of hard, nonlinear equality and inequality constraints [12]. Thus we can reframe the optimization in Eq. (4) to be:

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{N_c} g^2(x_k, \boldsymbol{\theta}) \quad \text{s.t.} \quad h_j = 0 \quad \forall j \in [1, p] \tag{5}$$

where h_j are the p-boundary/initial value *violations*, which should ideally be zero if g is the solution to the ODE. First a potential solution g is generated by normal GP crossover/mutation, and then its loss Eq. (5) is minimized by adjusting $\boldsymbol{\theta}$, the vector of embedded tree constants. Here the initial/boundary values are imposed as *hard* constraints via the constant tuning rather than as soft penalty terms as in Eq. (4).

(For comparison purposes, we have also solved the ODEs by minimizing Eq. (4), where we could maybe have reduced the runtime by using, for example, the low-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm. Instead, we used

algorithm—here quantities such as the population size. We use the term *constant tuning* here, however, to align with what we believe to be common usage in the GP literature.

[2] We have used the SLSQP algorithm from the NLOpt library (https://nlopt.readthedocs.io/en/latest/).

the SLSQP algorithm in order to make the different methods as comparable as possible.)

Third, the present work also rests on the ability to algorithmically differentiate a GP tree with respect to x. Given a tree f, we have presented a set of transformations [17] that can generate a sequence of separate trees that evaluate f', f'', f''', etc. This transformation is correct by construction, and is facilitated by all trees being guaranteed to be $\in \mathcal{C}^\infty$. It is thus straightforward to evaluate the loss functional Eq. (5) by combining the outputs of the requisite set of trees. This is a numerical advantage since the evaluation of derivatives, especially higher-order derivatives by automatic differentiation can be problematic [18]. (We do, however, elsewhere use automatic differentiation (AD) to evaluate only the first partial derivatives of the tree with respect to its parameters values.)

A remaining problem in this research is how to handle trees which, despite parameter tuning, are still unable to meet some or all hard problem constraints (within the user-specified tolerance). The subject of constrained optimization in evolutionary computing has received a huge amount of attention—see Rahimi et al. [19] for a recent review—with a very widely-used technique being the so-called *death penalty* whereby any individual violating even one of multiple constraints by even a marginal amount is assigned an infinite loss value, and is thus highly likely to be rapidly lost from the population. Despite its widespread use, the death penalty has been much criticized [19] since it does not differentiate between individuals that miss all of the constraints by huge margins and those that miss a single constraint by a minuscule amount. To address this issue, in this paper we propose what we believe to be a novel modification to the multiobjective ranking procedure used for sorting the population whereby comparison of two infeasible individuals assigns a higher rank to the tree that produces the smallest constraint violations.

1.1 Research questions

Given three possible methods of solving ODEs using GP—the soft penalty of Eq. (4), and two hard constraint implementations for solving Eq. (5)—we can identify a number of research questions, which we seek to address in this paper.

a) **RQ1:** For each of the three methods, how dependent are the solutions on the numbers of collocation points? Do, for example, few points lead to spurious interpolations between collocation points, and therefore overfitting?
b) **RQ2:** For a given differential equation and number of collocation points, are there any differences between the penalty and the two hard-constraint approaches?
c) **RQ3:** It is known that many DEs cannot be solved in terms of elementary mathematical functions. How well can a restricted set of GP functions approximate the solutions to ODEs?

1.2 Contributions of this paper

In this paper, we present a collocation-based GP method for approximating the solution of DEs, the contributions of which are:

i) We extend the previous work [10] on minimizing the soft penalty loss functional Eq. (4) to enforce *hard* constraints, which we achieve by introducing these into the numerical optimization of the tree constants.
ii) We introduce what we believe is a novel, multiobjective method of enforcing constraints in evolutionary computing whereby infeasible individuals are ordered by the degree to which they violate the constraint(s).
iii) We make a statistical comparison of: the novel hard constraint ordering method, the penalty method, and death penalty approaches.
iv) We further characterize the phenomenon previously observed in [10] whereby some ODEs produce trivial (i.e. $y = 0$) solutions, and explore the mitigation of this undesirable effect.

Table 1 Benchmark ordinary differential equations used in this work, after [3]

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	$y' = -y\cos(x)$	$\forall x \in [0, 1]$
		$y(0) = 1$
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	$y'' = -\frac{y'}{5} - y$	$\forall x \in [0, 2]$,
	$-\frac{1}{5}e^{-\frac{x}{5}}\cos(x)$	$y(0) = 0,\ y'(0) = 1$
ODE-10	$y'' = 4y' - 4y + e^x$	$\forall x \in [0, 1]$
		$y(0) = 3,\ y'(0) = 6$

The second column shows the domain and the initial condition(s)

2 Method

2.1 Benchmark differential equations

We have used the set of benchmark ordinary differential equations (ODEs) previously studied by Tsoulos and Lagaris [2], and by Seaton et al. [3], and detailed in Table 1. In this table, $y = y(x)$ denotes the solution, $y' = dy/dx$ and $y'' = d^2y/dx^2$.

For what follows, we note that, notwithstanding the initial conditions, ODE-4, ODE-6, ODE-7 and ODE-8 all have the possible trivial solution of $y = 0$.

2.2 Differential equation formulation

Probably the most convenient way of explaining the problem setup is by example. Taking, say, ODE-10, suppose we have a possible (ansatz) solution *f* in the form of a GP tree. Since ODE-10 is given by:

$$y'' - 4y' + 4y - e^x = 0$$

we need to evaluate the loss function, either Eqs. (4) or Eq. (5) via a series of steps:

1. Given tree *f*, algorithmically differentiate *f* to yield a completely separate tree f' that evaluates the first derivative of *f* w.r.t. *x* [17]. One of the potential advantages of calculating the loss using separate derivative trees, aside from avoiding the numerical issues that can sometimes result when calculating higher-order derivatives using AD [18], is that it facilitates the use of multiple threads to evaluate the overall loss value. For the time being, however, we leave this possibility for future work.
2. Apply the differentiation transformation a second time to f' to produce another separate tree f'' that evaluates the second derivative of *f* w.r.t. *x*.
3. For each of the N_c collocation points located at $x_i \in [a, b]$ where $i \in [1, N_c]$, evaluate the loss functional as: $g_i = f''(x_i) - 4f'(x_i) + 4f(x_i) - \exp(x)$, and sum the square of each contribution over *i*. In practice, g_i was evaluated as a Clifford dual number [5] using automatic differentiation to simultaneously yield both its value and a vector of partial derivatives of the loss w.r.t. the tree's parameters (constants).
4. The loss functional is minimized in order to tune the tree's constants subject to the initial/boundary constraints using the SQP algorithm. The initial conditions are incorporated either as a soft penalty term in Eq. (4), or as a hard constraint in Eq. (5). For Eq. (4), we have used a fixed value of $\lambda = 1$ since this is a popular choice in the DNN literature, and there is no principled way of selecting an optimal value [6]. The minimum objective value found by the SQP optimizer constitutes one of the tree's vector of fitness objectives; the tree's node count comprises the other fitness measure.

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 $\in [16 \ldots 128]$
Function set	Unary minus, $+$, $-$, $\times$, Analytic Quotient (AQ) [15]
Terminal set	Input variable and mutable constants
Initial mutable constants	Uniformly selected $\in \{0.1, 0.2, \ldots, 0.8, 0.9\}$
Objectives to minimize	i) Loss functional value from either Eq. (4) or Eq. (5), and ii) Tree node count
SQP convergence	50 iterations or objective $\leq 10^{-4}$
Boundary/initial value tolerance	10^{-6}
Selection	Pareto ranking – see [21]
No. of children	Two children produced per breeding operation
Crossover	Point crossover; probability $\Pr = 0.9$ of selecting an internal node
Crossover probability	1.0
Mutation	Subtree mutation [22, p.16] (full trees of depth 4)
Mutation probability	1.0
Total number of generated children	10,000

2.3 Genetic programming framework

The basic evolutionary algorithm used in this work is very similar to that employed previously [10, 11]. The parameter settings of this multiobjective genetic programming (MOGP) algorithm are shown in Table 2. We used a steady-state, as opposed to generational, evolutionary algorithm since our experience is that this gives superior results—see [20] and the discussion therein for more details. We employed a bi-objective fitness vector comprising: i) the value of the appropriate 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 (allowing tied ranks), and mapping each individual's rank to a scalar value with a linear function such that the best-ranked individuals received the largest value and the worst ranked received zero. Selection of the parents was then performed stochastically biased by these scalar values—see [21, p.32] for full details.

Each run of the GP program was allowed to generate 10,000 offspring, and the tree constants optimized in every child as outlined above, The tree with the smallest value of loss functional was taken as the best-of-run individual regardless of tree size, which is appropriate here since we are interested only in the best ODE solution. We generated a set of 30 different initial populations, and ran each GP setup once for each population.

Restriction of the function set to: unary minus, +, −, × and the AQ operator—see Table 2—was quite deliberate. Arguably, the evolutionary search could have been simplified by including functions such as sin and cos, but we were keen to avoid the criticism that can be leveled at Tsoulos and Lagaris [2] of implicitly embedding the exact solutions in the problem formulation. In practice, very large numbers of differential equations have no known solution in terms of elementary functions, and the solutions need to be synthesized. We have thus limited the function set to explore the ability of GP to compose such solutions for which there is a sound theoretical justification. The Weierstrass approximation theorem [23, §7.26] states that any continuous function can be uniformly approximated by a polynomial so, in principle, the function set employed here can synthesize the solution to any ODE.

In terms of RQ3 (Sect. 1.1), it is noteworthy that none of the (known) solutions of the benchmark ODEs in Table 1 can be exactly represented by the function set in Table 2.

2.4 Novel constraint-violation ranking method

As pointed out in Sect. 1, the commonly used death-penalty approach to constrained optimization fails to differentiate between an individual that marginally misses one constraint and another that fails to meet all constraints, each by a large margin. This is potentially problematic in practice, especially in initial populations where randomly-generated individuals are unlikely to meet many, or indeed any, of the constraints, even after constant tuning. We have observed cases where the death penalty approach produces initial populations where fewer than ∼5% of the initial population are feasible, which might be considered to severely restrict the genetic diversity of the population, and therefore search efficiency.

We have designed a scheme for comparing two individuals $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ allowing for the fact that: neither, only one, or both individuals can be infeasible. We characterize an individual's performance with a 2-tuple $(\boldsymbol{B}_i, \boldsymbol{C}_i)\, i \in [1, 2]$ where $\boldsymbol{B}_i$ itself is the 2-tuple comprising the value of loss functional and tree node count. For a problem with p constraints, $\boldsymbol{C}_i$ is a p-tuple comprising the values of each of the constraint *violations*. Without loss of generality, we consider only equality constraints so an individual is said to be *feasible* iff $|c_j| \leq \epsilon_c \ \forall j \in [1, p]$, and ϵ_c is a user-defined, non-negative tolerance on meeting the constraint. Throughout this work we have used a value of $\epsilon_c = 10^{-6}$.

We can devise a straightforward scheme to order $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ as follows that can be used in the population sorting:

i) If both $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are feasible then order $\boldsymbol{A}_{1,2}$ with conventional Pareto ranking on $\boldsymbol{B}_{1,2}$, the loss/node-count tuples.
ii) If $\boldsymbol{A}_1$ is feasible but $\boldsymbol{A}_2$ is infeasible then $\boldsymbol{A}_1$ precedes $\boldsymbol{A}_2$
iii) If $\boldsymbol{A}_1$ is infeasible but $\boldsymbol{A}_2$ is feasible then $\boldsymbol{A}_2$ precedes $\boldsymbol{A}_1$
iv) If $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are both infeasible then order $\boldsymbol{A}_{1,2}$ using conventional Pareto ranking on $\boldsymbol{C}_{1,2}$, namely, ranking the *constraint violations* of $\boldsymbol{A}_{1,2}$. Notice that the $\boldsymbol{B}_{1,2}$ tuples are ignored in this case.

Note that step (iv) overcomes the objection to the death penalty by ranking more highly the individual that violates the constraints by the *least* amount. This last stage also differentiates our approach from that of Mezura-Montes et al. [24] who summed the constraint violations and then ranked the two individuals using normal scalar comparison; our use of Pareto ranking assures that all constraint violations are treated equally and independently of scale. Our ranking scheme thus provides an evolutionary pressure to first make individuals feasible, and only then optimize the objective performance (of the now feasible solutions). Within the constraint-handling taxonomy of Rahimi et al. [19, §3.2.1], it is an *exterior* method.

Throughout, we refer to GP setups employing this novel style of population ranking as the *hard constraint* (**hc**) method. This is distinct from the *death penalty* (**dp**) method, which, strictly, also imposes a hard constraint, but does so in an unsatisfactory way.

2.5 Trivial solutions

We noted in Sect. 2.1 that ODE-4, ODE-6, ODE-7 and ODE-8 all allow trivial (i.e. $y = 0$) solutions notwithstanding the initial conditions, and many runs for these ODEs did indeed produce trivial solutions. Figure 1 shows a typical example of the relative error plot of a trivial solution to ODE-4 solved using the **hc** method for 10 collocation points. The relative error is very small ($\sim$ zero) at the point where it matches the initial condition—indicating accurate satisfaction of the initial value—but rapidly falls to a value of -1, namely 100% in error thereafter; although the plot appears to be a discontinuous step function on the scale shown in the figure, it is actually constrained by the GP setup used here to be a smooth curve. Such functions are manifestly *not* solutions of the given ODE. Increasing the numbers of collocation points or changing the constraint-handling method (**hc**, **dp** or **pm**) had no effect on this behavior.

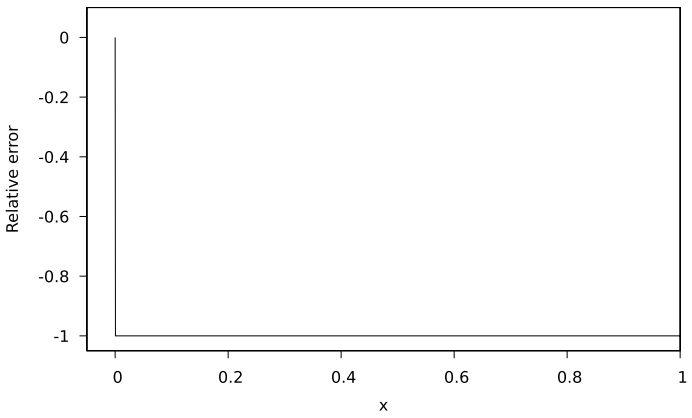


Fig. 1 An example of a trivial solution generated by solving ODE-4 using the hard-constraint method with 10 collocation points

In order to combat this undesired behavior, where necessary, we have supplemented the loss functions for all three methods (**hc**, **dp** and **pm**) by adding an additional constraint on the numerical optimization given in Eq. (6).

$$\sum_{k=1}^{N_c} f^2(x_k, \boldsymbol{\theta}) - T \geq 0 \tag{6}$$

where f is the solution indicated by the GP tree, and T is a non-negative, user-defined threshold. This is motivated by the observation that the trivial solution is very attractive to the optimizer, something that is borne out by extensive studies of the loss landscape in the DNN literature—see, for example, Rohrhofer et al. [25] for a particularly elegant exposition. Consequently, we have added the further constraint Eq. (6) to make the trivial solution unattractive/infeasible. an approach entirely consistent with the classical numerical optimization approach of penalizing 'undesirable' regions of the domain [12]. In the same way that the mean-squared error term in the loss function favors solutions in a particular region of the space, the constraint Eq. (6) disfavors undesired solutions. In the case of the **hc** and **dp** methods, this additional constraint has been added as a further hard inequality constraint supplied to the SLSQP optimizer. In the case of the **pm** method, the extra constraint has been added to the soft penalty term P in Eq. (4).

An alternative view of Eq. (6) is to impose an extra constraint such that the 'energy' in the evolved function is non-zero while also accommodating the situation where the function might genuinely be required pass through zero at some of the collocation points. We experimented with various values of T and finally settled on half the value of test error for the trivial solution for any given ODE, but the exact value did not appear too critical providing it was greater than zero. The use of this additional constraint in this context has previously been reported in [10].

Somewhat more complicated remedies for the trivial solution problem have been proposed in the DNN literature—so-called *time marching* approaches—that involve disjoint partitioning the domain into smaller, manually-determined segments such that the deep basin of attraction in the loss function around the trivial solution is either removed or diminished. These time-marching methods have been reviewed and unified by Penwarden et al. [26]. The main drawback of time-marching approaches is that the solution for the i-th partition—and any errors contained in it—is propagated on to form the initial condition for the $(i+1)$-th partition, and so on. The scheme, therefore, potentially suffers from an accumulation of errors.

2.6 Statistical testing

In this work we have compared the hard-constraint (**hc**), death-penalty (**dp**) and penalty-method (**pm**) approaches for each of the set of benchmark ODEs with a variety of numbers of collocation points. We followed the usual approach in the GP literature of pairing based on starting from the same initial population, repeating each set of experiments thirty times, each with a different, independent initial population. These thirty sets of paired results were then analyzed for statistical difference.

In particular, we tested the zero-difference null hypotheses of: i) the differences in test error, ii) the differences in absolute initial value violations, and iii) for the 2nd-order equations, the differences in absolute initial derivative value violations[3].

We have calculated the test error of an individual as the sum of squared differences between the evolved and known analytical solutions over 2000 equally-spaced points over the domain. We are thus probing the quality of the solution *between* the collocation points where the solution is reliant on generalization, and an overfitted solution might be presumed to exhibit a spurious fit.

As with previous work [10, 11], we have tested the statistical differences using a Bayesian Wilcoxon signed-rank test [28] for a range of reasons[4]. Firstly, traditional frequentist null-hypothesis statistical testing (NHST) has received a great deal of criticism, and is open to widespread misinterpretation [29, 30]. Secondly, since NHST makes yes/no decisions (based on significance levels), its repeated use requires multiple-test corrections, such as the Bonferroni procedure or similar, which can require intricate assumptions. Bayesian testing, on the other hand, requires no such multiple-test corrections, which is a significant technical advantage here where we are making a large number of comparisons. Further, Bayesian testing can also provide evidence in favor of the null hypothesis unlike NHST which only facilitates a far weaker statement that the null hypothesis cannot be rejected.

Traditionally, when faced with a number (>2) of groups, a frequentist analyst would perform a Friedman test, and then carry-out pairwise testing if the Friedman test suggested inhomogeneity. We suspect this route is followed for two reasons: firstly, computational, which is hardly a valid reason nowadays. Second, in order to minimize the total number of comparisons given that frequentist tests require multiple-test corrections.

In the present work, we dispense with this Friedman-like step because a Friedman test is simply a generalization of the Wilcoxon test to more than two groups so a repeated set of Wilcoxon tests is equivalent to a single Friedman test. Given that the computational burden is not a factor, it is simpler to adopt a one-stage as opposed to a two-stage testing procedure.

As to how to interpret the results of a Bayesian Wilcoxon tests, since these report the posterior probability that the tested statistic is greater/less than zero, we adopt the guidelines of Kass and Rafferty [31, Section 3.2], and regard any posterior probability falling *outside* the interval [0.091, 0.91] as providing "strong" evidence of statistical difference one way or the other, since this implies that one outcome is at least ten times more probable than the alternative.

Notwithstanding, any finding of statistical significance from a hypothesis test does not necessarily imply any meaningful difference between the two groups [32]. To quote Cohen's famous aphorism, "The primary product of a research inquiry is one or more measures of effect size, not P values" [33]. Consequently, we have also estimated a number of *effect sizes* by computing the 5%-to-95% confidence intervals (CIs) of the differences using bias-corrected and accelerated bootstrap

[3] In this section, we have attempted to follow the Bayesian analysis reporting guidelines of Kruschke [27].

[4] We have used version 1.0.3 of the baycomp Python implementation of this test available from https://pypi.org/project/baycomp/.

resampling [34].[5] As to how to interpret effect sizes, these are clearly application-dependent. In the present work, we have taken the interval $[-10^{-6}, +10^{-6}]$ as the *region-of-probable-equivalence* (ROPE) [32] within which we consider the differences to be inconsequential. This is approximately the precision of a single-precision IEEE754:2019-compliant floating point number, and is equivalent to saying that if two possible models give identical predictions to the sixth decimal place then the difference between them is of no practical importance.[6] If the confidence interval (CI) on a measure is enclosed inside the ROPE interval then any difference is judged of no interest regardless of the outcome of the statistical test. Further, if the CI includes the zero value then that effect size is also unimportant as the null hypothesis cannot be credibly discounted.

3 Results

In order to address the research questions in Sect. 1.1, we have conducted a number of statistical analyses. To address RQ1, for a given method (**hc**, **dp** or **pm**) and a given ODE, we statistically compared the results for each of: 10, 20 50 and 100 collocation points against each other. As explained in Sect. 2.6, there is no impediment to this set of exhaustive comparisons (other than the computing time required). Since the total volume of statistical results is too large to include in the main body of this paper, we report in Sect. 3.1 only an interesting/representative subset; the Supplementary Material accompanying this paper contains the complete set of results.

To address RQ2, Sect. 3.2 compares the three evolutionary methods: **hc**, **dp** and **pm** for each ODE and for each number of collocation points.

As with Sect. 3.1, we reproduce only a subset of typical results; the reader can again find the full set of results in the Supplementary Material to this paper.

While the statistical testing described above provides information on the relative performance of different methods 'on average', we also include in Sect. 3.3 a selection of the 'best' results we have obtained for a range of benchmark ODEs as these will be the results of greatest interest in a practical application. These were all selected on the basis of yielding the smallest training error rather than the best test performance since this is more realistic in the practical situation where the true ODE solution is unknown. The research question (RQ3) as to whether the restricted function set used in this work is able to synthesize accurate solutions is also addressed by Sect. 3.3.

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

[6] Readers may prefer a different ROPE. The full set of statistical test data is provided in supplementary material to this paper to allow readers to gauge the impact of a different choice of ROPE.

Table 3 Comparisons of numbers of collocation points for ODE-1 for penalty-method

		No. of collocation points					
		20		50		100	
No. of collocation points	10	0.124	0.528	0.067	0.428	0.142	0.308
		− 1.16e−9	− 6.01e−5	− 5.07e−9	− 8.78e−5	− 2.80e−9	− 3.76e−5
		2.69e−10	4.95e−6	3.10e−9	7.28e−6	2.02e−9	3.02e−6
		1.06e−7	4.12e−5	1.42e−7	1.26e−4	1.28e−7	7.95e−5
	20			0.231	0.270	0.272	0.205
				− 1.53e−8	− 1.86e−5	− 5.27e−9	− 7.26e−5
				3.35e−9	5.63e−6	− 1.04e−19	4.43e−6
				1.33e−7	2.27e−4	1.06e−7	1.72e−4
	50					0.341	0.285
						− 3.12e−9	− 4.13e−5
						3.69e−14	8.87e−7
						1.88e−8	8.42e−5

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

Table 4 Comparisons of numbers of collocation points for ODE-5 for penalty-method

		No. of collocation points					
		20		50		100	
No. of collocation points	10	0.0	0.995	2e-5	0.995	0.0	0.995
		1.85e−12	− 7.11e−9	2.77e−12	− 6.59e−9	2.18e−12	− 4.83e−9
		7.30e−12	− 3.35e−9	7.41e−12	− 1.29e−9	6.99e−12	− 1.90e−9
		1.29e−11	− 6.23e−10	1.09e−11	− 2.42e−10	1.18e−11	− 5.37e−10
	20			0.099	0.429	0.557	0.589
				− 8.03e−13	− 3.76e−9	− 7.17e−13	− 3.95e−9
				4.87e−13	− 6.14e−10	− 8.66e−14	− 2.84e−10
				2.17e−12	4.30e−9	6.45e−13	3.36e−9
	50					0.941	0.512
						− 1.43e−12	− 1.73e−9
						− 4.17e−13	2.38e−10
						1.99e−13	4.81e−9

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

3.1 Influence of the numbers of collocation points

3.1.1 Penalty method (pm)

We show the comparisons for varying numbers of collocation points for ODE-1 (Table 3), ODE-5 (Table 4) and ODE-10 (Table 5) as typical of this group of results. To take Table 3 as an example, the first group of three rows shows the comparison of 10 collocation points against 20, 50 and 100 points, respectively. The first entry in each column-wise group shows the posterior probability that 10 collocation points yield a better test error than 20, 50 or 100 points while the second entry shows the

Table 5 Comparisons of numbers of collocation points for ODE-10 for penalty-method

	No. of collocation points	No. of collocation points								
		20			50			100		
No. of collocation points	10	0.0	0.221	0.597	0.0	0.715	0.817	0.0	0.472	0.525
		7.40e−11	− 4.39e−9	− 2.08e−9	1.14e−10	− 4.74e−9	− 7.00e−9	1.57e−10	− 5.98e−9	− 3.98e−9
		3.30e−10	− 5.27e−10	3.00e−9	3.73e−10	− 1.24e−9	− 1.76e−9	3.68e−10	6.60e−10	1.94e−9
		7.80e−10	4.28e−9	9.24e−9	7.77e−10	3.69e−9	8.49e−9	7.82e−10	4.77e−9	5.05e−9
	20				0.006	0.938	0.889	8e-5	0.724	0.388
					− 1.29e−12	-4.97e−9	− 9.93e−9	7.37e−13	− 2.91e−9	− 9.73e−9
					8.31e−12	− 1.13e−9	− 9.64e−10	1.67e−11	− 3.33e−10	− 3.96e−10
					4.36e−11	− 9.03e−11	1.00e−9	7.05e−11	5.29e−9	1.86e−9
	50							0.002	0.503	0.346
								3.22e−14	− 5.36e−9	− 1.24e−8
								2.36e−12	5.97e−10	− 5.03e−10
								1.04e−11	6.76e−9	6.35e−9

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

posterior probability that 10 points yields a smaller initial value violation than 20 points, and so on. Specifically, the probability that 10 points has a better test error than 20 points is 0.124, while the corresponding figure for the 10-versus-50 comparison is 0.067. Similarly, the 5%-95% confidence interval (CI) for the 10-versus-20 test error comparison is $[1.16 \times 10^{-9}, 1.06 \times 10^{-7}]$ with a median difference of 2.69×10^{-10}. The posterior probability that 10 points has a better initial value violation that 20 points is 0.528. The corresponding CI for the 10-versus-20 initial value violations is $[-6.01 \times 10^{-5}, 4.12 \times 10^{-5}]$ with a median of -4.95×10^{-6}. The second-order ODEs, of course, have an additional constraint on the value of the first-derivative, which is given, where appropriate, as a third column group; for Table 5, for example, the probability that 10 points has a better first-derivative initial value violation that 20 points is 0.597. Throughout this paper, we have used this exact same table layout consistently. (A more detailed description of the table layouts is given in Sect. S4 of the Supplementary Material.)

Taken as a whole, the results in this **pm** sub-section display remarkably little evidence of any difference in either the test errors or the initial value violations for varying the numbers of collocation points.

In terms of the four ODEs discussed in Sect. 2.1 that allow trivial solutions, and the mitigation proposed in Sect. 2.5, the results are mixed. ODE-6 produced all non-trivial solutions without any need for the additional threshold constraint in Sect. 2.5; in fact, ODE-6 failed to produce a single trivial solution for any method or any number of collocation points. The influence of the threshold constraint for the **pm** method, however, for ODE-4, ODE-7 and ODE-8 is summarized in Table 6 from which it is clear that while its inclusion is both necessary and effective for ODE-4 and ODE-8, its influence on ODE-7 is very limited.

3.1.2 Death penalty method (dp)

We again show a selection the interesting statistical results for the death penalty method for: ODE-1, ODE-5, ODE-7 and ODE-10. The results for ODE-7 are for the variant with the extra threshold constraint only since without this constraint, all runs produced trivial solutions (Tables 7 and 8).

Most of the results for the death penalty (**dp**) method display no evidence of statistical superiority for any number of collocation points apart from for ODE-7 (Table 9) and ODE-10 (Table 10) where the cells in the tables indicating statistical significance are shown in bold. For ODE-7, 10 collocation points appears to produce an inferior test error compared to 20, 50 and 100 points although neither of the initial conditions for this 2$^{\text{nd}}$-order equation suggests any difference. An identical observation can be made for ODE-10 (Table 10): 10 points again appears too few. None of the other ODEs produces any statistically-significant differences for the **dp** method apart from ODE-8 although see the discussion below.

In relation to the problematic subset of ODEs that can produce trivial solutions, again ODE-6 did not require the extra threshold constraint to produce exclusively non-trivial solutions. Table 11 summarizes the effect of adding the threshold constraint for the **dp** method. Compared to Table 6, the threshold constraint here was much more effective for ODE-7 with all thirty runs producing non-trivial solutions

Table 6 Numbers of converged runs from the 30 repetitions for ODE-4, ODE-7 and ODE-8 with and without the additional threshold constraint for varying numbers of collocation points for the penalty method

ODE	# Collocation pts.			
	10	20	50	100
ODE-4 without	10	19	25	23
ODE-4 with	30	30	30	30
ODE-7 without	0	0	0	0
ODE-7 with	0	1	1	6
ODE-8 without	0	0	4	9
ODE-8 with	30	30	30	30

Table 7 Comparisons of numbers of collocation points for ODE-1 death-penalty for hard-constraints

		No. of collocation points					
		20		50		100	
No. of collocation points	10	0.250	0.393	0.619	0.808	0.463	0.714
		− 1.44e−29	− 7.11e−15	− 8.76e−23	− 2.12e−12	− 1.23e−22	− 2.52e−12
		8.99e−31	0.00e+0	− 1.34e−31	− 1.78e−15	4.23e−31	− 1.78e−15
		2.51e−18	9.59e−14	6.99e−26	1.78e−15	1.40e−25	7.11e−15
	20			0.660	0.790	0.140	0.688
				− 3.02e−21	− 1.25e−12	− 3.46e−31	− 7.11e−15
				− 3.94e−31	− 1.78e−15	4.74e−31	0.00e+0
				2.36e−18	1.78e−15	3.49e−17	5.33e−15
	50					0.713	0.788
						− 9.42e−18	− 4.81e−11
						3.74e−31	0.00e+0
						1.34e−28	3.55e−15

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

Table 8 Comparisons of numbers of collocation points for ODE-5 death-penalty for hard-constraints

		No. of collocation points					
		20		50		100	
No. of collocation points	10	0.0	0.928	0.0	0.997	0.0	0.985
		3.30e−12	− 1.02e−11	3.25e−12	− 2.46e−11	3.39e−12	− 1.64e−11
		1.05e−11	− 2.32e−12	9.89e−12	− 7.86e−12	1.06e−11	− 4.22e−12
		2.80e−11	4.13e−13	2.82e−11	− 2.10e−12	2.82e−11	− 4.63e−13
	20			0.0156	0.801	0.002	0.230
				6.37e−16	− 1.50e−11	3.29e−14	− 3.97e−12
				8.06e−14	− 4.06e−12	1.00e−13	6.76e−13
				2.20e−13	1.69e−12	2.14e−13	1.59e−11
	50					0.098	0.080
						− 6.77e−15	− 3.10e−12
						1.50e−14	5.52e−12
						6.36e−14	2.26e−11

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

Table 9 Comparisons of numbers of collocation points for ODE-7 death-penalty with threshold for hard-constraints

		No. of collocation points								
		20			50			100		
No. of collocation points	10	**0.007**	0.955	0.091	**0.0**	0.421	0.703	**8e-5**	0.374	0.385
		1.82e−2	− 1.46e−10	− 8.53e−14	**4.22e−2**	− 1.26e−7	− 1.25e−20	**2.80e−2**	− 1.04e−9	− 1.44e−19
		2.53e−1	1.05e−10	− 1.55e−18	**2.81e−1**	− 1.69e−14	0.00e+0	**1.60e−1**	1.51e−14	0.00e+0
		3.45e−1	7.97e−8	1.98e−21	**3.13e−1**	6.56e−8	1.25e−17	**3.40e−1**	3.88e−8	1.57e−17
	20				0.570	0.066	0.980	0.821	0.083	0.723
					− 1.63e−2	− 1.26e−7	− 2.01e−22	− 2.17e−2	− 3.36e−8	− 6.64e−24
					− 1.82e−3	− 1.66e−9	4.72e−19	− 1.49e−2	− 1.77e−11	1.07e−18
					1.08e−2	9.77e−15	3.66e−12	5.14e−3	1.14e−8	4.26e−14
	50							0.920	0.161	0.194
								− 2.05e−2	− 4.54e−10	0.00e+0
								− 1.12e−2	7.67e−10	0.00e+0
								4.71e−3	7.07e−8	2.44e−15

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

for all numbers of collocation points. The effect on ODE-8, however, was more limited than for the **pm** method—as a consequence, the raw statistical comparisons for ODE-8 in the Supplementary Material need to be treated with caution as some of the comparisons are with trivial solutions and therefore of questionable validity.

3.1.3 Hard constraint method (hc)

For the hard constraint (**hc**) method we present results for ODE-1, ODE-5, ODE-7 with threshold, and ODE-8 with threshold, these last two being the only data that indicate any statistically-significant differences; again, the cells containing significant results are shown in bold (Tables 12 and 13).

For ODE-7 with threshold (Table 14), 10 collocation points appears too few in terms of test error, but again not initial value violations. One hundred points appears superior to 20 collocation points, but 50 and 100 appear equivalent. For ODE-8, 10 collocation points also appears too few. There are no other statistical differences in this set of comparisons.

Table 16 provides a summary of the effectiveness of the additional threshold constraint for the **hc** hard constraint method. Whereas the threshold constraint was only partially successful in suppressing trivial solutions for the **dp** and **pm** methods, for the hard constraint (**hc**) method all the solutions are non-trivial for all ODEs (Table 15).

3.2 Comparison between methods

The data presented in this section are designed to address RQ2, whether any of the three methods is superior (Tables 17, 18, and 19). Unlike the previous section where we observed a few instances where 10 collocation points appears to be too few but mostly no statistical differences, a direct comparison between methods yields many more statistical-significant instances. We present only a subset of the tabular data, all only for 50 collocation points, together with a summary of the method comparisons for all numbers of collocation points in Table 21; as before, the cells containing statistically significant results are shown in bold. The complete set of data are provided in the Supplementary Material.

Table 21 shows a summary of the method comparisons—no entry indicates no evidence of statistical difference. As for the entries in this table, the predecessor relation '$\prec$' denotes, with a slight abuse of notation, statistical superiority of the method on the left since smaller is better for all measures. Thus, for a given number of collocation points, **hc** $\prec$ **pm** indicates that the hard constraint method is better than the penalty method, and so on. The entry '**hc** $\prec$ **pm**, **dp** $\prec$ **pm**' (note the comma) denotes that **hc** is superior to **pm**, *and* **dp** is superior to **pm**, but that **hc** is *not* superior to **dp**. On the other hand, '**hc** $\prec$ **dp** $\prec$ **pm**' denotes that **hc** is superior to **dp** which is in turn superior to **pm**. This combination only occurs for ODE-7—see, for example, Table 20.

Overall, the sparsity of Table 21 indicates that for most comparisons, the three methods appear mostly equivalent on all metrics. Interestingly, ODE-1 and ODE-2 suggest that the penalty method is worst at meeting the initial conditions, while for

Table 10 Comparisons of numbers of collocation points for ODE-10 death-penalty for hard-constraints

		No. of collocation points								
		20			50			100		
No. of collocation points	10	**0.0**	0.032	0.031	**0.000**	0.018	0.0180	**0.000**	0.090	0.089
		6.10e−11	0.00e+0	0.00e+0	**3.64e−11**	0.00e+0	0.00e+0	**6.14e−11**	0.00e+0	0.00e+0
		6.86e−10	0.00e+0	0.00e+0	**2.77e−10**	0.00e+0	0.00e+0	**6.92e−10**	0.00e+0	0.00e+0
		1.28e−6	4.44e−16	4.44e−16	**1.28e−6**	6.66e−16	6.66e−16	**1.28e−6**	2.22e−16	2.22e−16
	20				0.033	0.498	0.500	0.026	0.944	0.944
					2.39e−14	0.00e+0	0.00e+0	9.50e−14	0.00e+0	0.00e+0
					1.15e−12	0.00e+0	0.00e+0	1.62e−12	0.00e+0	0.00e+0
					4.16e−12	0.00e+0	0.00e+0	3.98e−12	0.00e+0	0.00e+0
	50							0.623	0.819	0.820
								− 3.86e−13	0.00e+0	0.00e+0
								− 6.50e−14	0.00e+0	0.00e+0
								3.40e−13	0.00e+0	0.00e+0

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

ODE-2, ODE-4_t, ODE-7_t and ODE-8_t there are also differences in the test error results. Regarding the test error, sometimes **pm** is better than **dp** (ODE-4, ODE-8), other times **dp** is superior to **pm** (ODE-2, ODE-7). In all the cases where there is a statistical difference, the hard constraint (**hc**) method is either superior to the **dp** and/or **pm** methods but never inferior to either.

3.3 Best-of-runs results

Although the preceding sections consider the performance of the competing methods 'on average', the best-of-run individuals are of greatest interest in practice. In this section we therefore show a set of individuals which are best-in-run selected on the basis of smallest training error for the hard constraint (**hc**) method only since it is either superior to but never inferior to the **dp** or **pm** methods. As with other results, the full set of best-of-run individuals for all methods is available in the Supplementary Material. In viewing the best-of-run results, however, a number of points need to be borne in mind:

1. The results here are the 'extremes' in the distributions of approximations so are not typical of a given method/number of collocation points. Nonetheless, they do give some indication of the bound on the accuracies that can be obtained.
2. In general, the solution for a given ODE will not be known so presenting the result that is closest to the known, analytical solution via the test error is misleading. The value of the technique in practice will be for ODEs for which there is no closed-form solution. We therefore present the results for the GP run which gave the smallest *training loss*, which may be considered unusual in the domain of machine learning where a more elaborate model selection routine is typically used. Here, however, we are not learning a function from (noisy) data, but rather solving an equation.
3. Finally, we present plots of *relative error* since this is the most useful practical measure, and effectively gauges the number of accurate digits in the GP approximation.

Since the results in this section are 'extreme' samples, there is unsurprisingly no consistent pattern of behavior across different methods, even when one method is statistically superior to another. There also does not appear to be any relationship between the method used or the number of collocation points, and the size of the relative errors.

Figure 2 shows the best-of-run plots for the first-order ODEs. The two most striking plots are Fig. 2a and f. Examination of the ordinate scales of these two plots reveals that the relative errors are made up of rounding errors in the double-precision floating point arithmetic used for the calculations. The remaining plots in Fig. 2 show the sort of characteristic oscillatory structure that is usual in function approximation.

Figure 3 shows the best-of-run results for the second-order ODEs. Most of the results are qualitatively similar to Fig. 2 apart from Fig. 3a, which exhibits 'spiking' behavior on top of a approximately zero baseline. On closer inspection, these spikes

Table 11 Numbers of non-trivial solutions from the 30 repetitions for ODE-4, ODE-7 and ODE-8 with and without the additional threshold constraint for varying numbers of collocation points for the death penalty

ODE	# Collocation pts.			
	10	20	50	100
ODE-4 without	0	6	5	8
ODE-4 with	30	30	30	30
ODE-7 without	0	0	0	0
ODE-7 with	30	30	30	30
ODE-8 without	0	0	0	0
ODE-8 with	24	26	26	26

Table 12 Comparisons of numbers of collocation points for ODE-1 for hard-constraints

		No. of collocation points					
		20		50		100	
No. of collocation points	10	0.881	0.986	0.887	1.000	0.948	1.000
		− 1.53e−14	− 1.46e−12	− 4.21e−17	− 5.18e−11	− 3.37e−16	− 1.27e−10
		− 1.05e−29	− 3.55e−15	− 1.58e−19	− 1.28e−12	− 7.22e−21	− 1.31e−13
		1.61e−29	0.00e+0	1.57e−30	0.00e+0	1.79e−30	− 1.78e−15
	20			0.642	0.830	0.683	0.984
				− 4.32e−14	− 1.64e−11	− 5.06e−18	− 7.42e−11
				− 1.58e−19	− 7.11e−15	− 8.87e−23	− 6.04e−14
				2.23e−20	3.89e−13	4.11e−23	0.00e+0
	50					0.588	0.329
						− 1.54e−15	− 3.57e−12
						1.86e−26	1.78e−15
						3.57e−17	4.30e−11

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

Table 13 Comparisons of numbers of collocation points for ODE-5 for hard-constraints

		No. of collocation points					
		20		50		100	
No. of collocation points	10	0.0	0.942	0.0	0.987	0.0	0.753
		2.59e−12	− 8.02e−12	2.04e−12	− 1.08e−11	2.66e−12	− 2.50e−12
		3.90e−12	− 3.21e−13	5.10e−12	− 1.90e−12	6.44e−12	− 5.06e−14
		1.04e−11	2.49e−13	1.17e−11	− 3.73e−14	1.16e−11	4.62e−13
	20			0.264	0.435	0.281	0.411
				− 3.41e−13	− 6.30e−12	− 9.13e−13	− 8.41e−13
				3.76e−13	5.85e−13	3.53e−13	6.75e−14
				5.88e−13	6.82e−12	1.08e−12	3.65e−12
	50					0.316	0.357
						− 7.09e−13	− 1.86e−12
						4.45e−13	− 5.06e−14
						8.97e−13	6.17e−12

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

Table 14 Comparisons of numbers of collocation points for ODE-7 with threshold for hard-constraints

		No. of collocation points								
		20			50			100		
No. of collocation points	10	**0.000**	0.918	0.716	**0.000**	0.810	0.967	**0.0**	0.694	0.975
		1.01e-3	− 1.78e−14	− 1.82e−20	**5.78e−4**	− 3.97e−11	− 1.21e−22	**1.74e−3**	− 9.92e−11	− 4.39e−23
		4.53e−2	0.00e+0	0.00e+0	**6.79e−3**	0.00e+0	0.00e+0	**4.53e−2**	0.00e+0	0.00e+0
		3.66e−1	0.00e+0	0.00e+0	**2.76e−1**	0.00e+0	0.00e+0	**3.74e−1**	0.00e+0	0.00e+0
	20				0.305	0.490	0.702	*0.006*	0.400	0.959
					− 4.31e−6	− 2.09e−11	− 1.21e−22	*1.28e−8*	− 9.92e−11	− 4.39e−23
					4.27e−7	0.00e+0	0.00e+0	*1.05e−6*	− 3.55e−15	0.00e+0
					2.37e−5	8.88e−15	1.82e−20	*8.39e−5*	0.00e+0	1.71e−18
	50							0.099	0.382	0.612
								− 3.01e−7	− 5.68e−14	0.00e+0
								3.74e−8	0.00e+0	0.00e+0
								9.64e−6	7.11e−15	6.22e−35

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

are nothing more than a consequence of the calculation of relative error. In this case, the *absolute* errors are in the range $\sim \pm 8 \times 10^{-5}$ across the whole domain, but at certain points, the value of the true function—the denominator in the relative error calculation—is very small leading to several large values of the quotient. So aside from the numerical artifacts, the relative error for this result is actually on the scale of 10^{-5}. (We return to analyze this result further in Sect. 4.)

Displaying the best-performing trees as expressions or graphs is not particularly helpful due to their complexity. We have, however, included a selection of best-performing trees for each ODE for 50 collocation points obtained using the hard constraint (**hc**) method in the Supplementary Material; these are formatted as human-readable Genetic Programming Markup Language (GPML) [35]. As is commonplace in GP, we make no claims that these trees are minimal implementations. Indeed, from inspecting them, it is quite clear that they contain significant amounts of redundancy (e.g. `constant + constant` terms).

4 Discussion and future work

An interesting and legitimate question is whether the GP search is finding simple polynomials as solutions to the ODEs, or fully exploiting its abilities as a universal approximator. The most straightforward way to explore this question is to see how well the ODEs' solutions can be represented by simple polynomials. We have generated least-squares fits with 4^{th}-, 6^{th}- and 8^{th}-order polynomials to the ODEs' analytical solutions over 50 uniformly spaced points in the domains; the largest absolute relative errors were calculated over 500 uniformly-spaced test points. We have limited the polynomial orders to 4, 6, and 8 in order to ensure a reasonable condition number on the least-squares estimations. In practice, one would not generally use algebraic polynomials here, but rather something like a Chebyshev polynomial for numerical reasons. However, since the question is whether GP is discovering simple polynomials, an algebraic polynomial is the object that GP is constrained to find so this is a fair comparison. The results are shown in Table 22.

From comparing Table 22 with Figs. 2 and 3 we conclude that the errors from even the 8^{th}-order polynomials are generally much worse than GP, especially for many of the second-order ODEs. The only polynomial results that come close to GP are for ODE-3 and ODE-4, but these are both quite smooth functions that one would expect to be well approximated by a low-order polynomial. There thus seems little evidence that GP is finding simple polynomials as solutions.

Additionally, inspection of the best-performing trees included in the Supplementary material shows that the analytic quotient (AQ) node is employed quite liberally at all depths in the generated trees implying that the functions synthesized by GP are not simple polynomials.

A notable conclusion from this research is that the different methods—**hc**, **dp** and **pm**—mostly show little variation in their test errors and ability to meet the initial/boundary value constraints. As has been noted, the summary of method comparisons in Table 21 is fairly sparse. Aside from ODE-1 and ODE-2, all methods without the complication of the threshold constraint appear statistically similar despite the

Table 15 Comparisons of numbers of collocation points for ODE-8 with threshold for hard-constraints

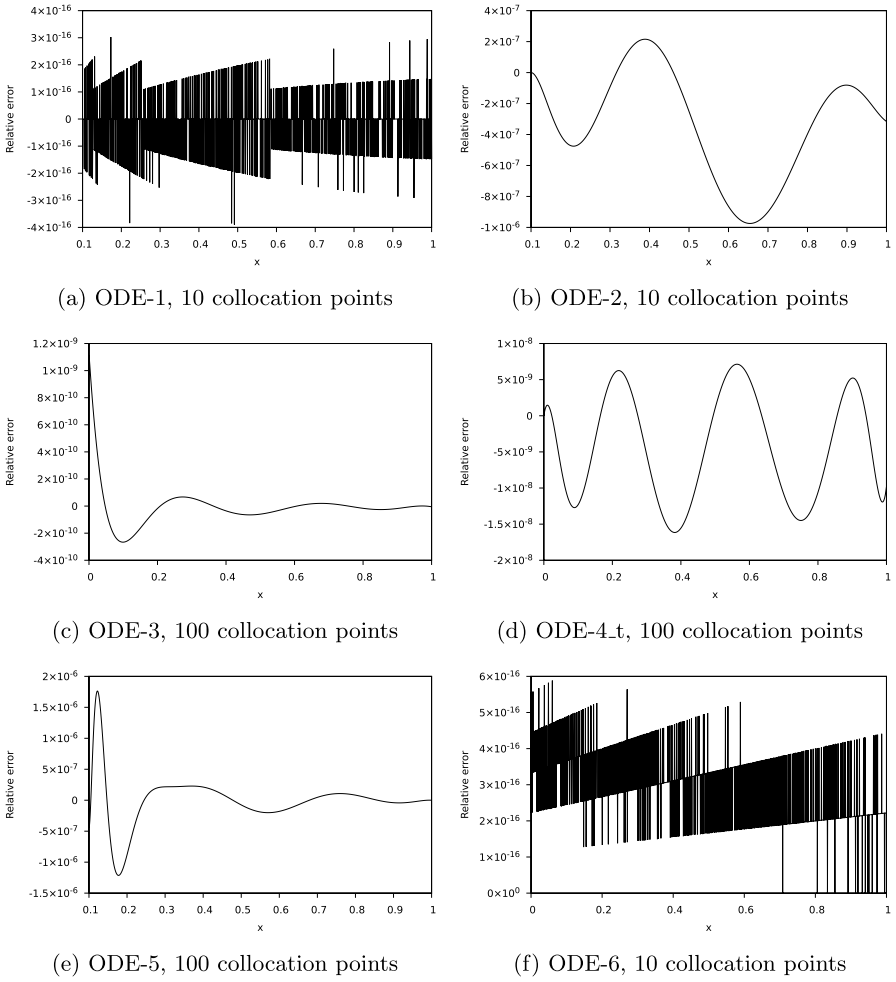
		No. of collocation points								
		20			50			100		
No. of collocation points	10	**2e-5**	0.911	0.986	**0.000**	0.554	0.967	**2e-5**	0.913	0.996
		2.24e−7	− 7.53e−13	− 1.75e−15	**2.18e−7**	− 2.20e−13	− 9.41e−26	**2.24e−7**	− 2.02e−12	− 3.61e−16
		1.06e−6	− 2.76e−14	− 7.10e−24	**1.78e−6**	− 1.79e−14	6.04e−36	**1.78e−6**	− 2.40e−14	− 4.26e−24
		6.33e−5	4.44e−16	5.83e−31	**6.33e−5**	− 4.44e−16	1.03e−29	**6.33e−5**	0.00e+0	5.82e−28
	20				0.019	0.305	0.250	0.114	0.140	0.560
					− 1.65e−13	− 1.82e−13	− 9.75e−26	− 3.21e−11	− 8.24e−13	− 2.21e−26
					3.88e−10	1.07e−14	1.99e−25	4.94e−11	2.22e−16	8.55e−24
					8.32e−10	6.11e−13	8.30e−22	5.52e−10	1.03e−13	1.75e−15
	50							0.446	0.394	0.711
								− 3.02e−11	− 7.84e−13	− 6.42e−24
								8.52e−12	− 1.11e−16	0.00e+0
								4.02e−11	7.11e−14	7.20e−26

Each cell contains the posterior probability that the row-wise entry is superior to the column-wise entry, together with the confidence interval for the same comparison. See Sect. 3.1.1 for an explanation of the table layout

obvious dangers of the death penalty method restricting genetic diversity, and the challenges of appropriately weighting the initial conditions in the soft penalty loss function Eq. (4). (Why ODE-1 and ODE-2 stand apart will be discussed below.) The obvious hypothesis is that the parameter tuning has a dominant effect on the performance of the three methods although this is difficult to test as parameter tuning is central to two (**hc** and **dp**) of the three methods considered. In previous work [10], we established that including parameter tuning made a significant difference to the performance of the penalty method. As discussed in Sect. 1, the soft penalty method should, in principle, yield some sort of trade-off between an accurate solution (the first term in Eq. (4)) and attainment of the initial conditions (the P term in Eq. (4)). This has received considerable discussion in the deep neural network (DNN) literature [6, 7] although including hard constraints in DNN training is computationally challenging due to the scale of the problem [14]. We conjecture, however, that the criticism of the soft penalty method is rather more nuanced. If we consider an ODE with a solution $f(x)$ over domain $x \in [a, b]$ subject to an initial condition of $f(a) = c$, an accurate approximation to f over the whole domain will also yield a good approximation to the initial value constraint. In short, in the case of differential equations, there is no trade-off, and the minimization of both terms in Eq. (4) are not in conflict. The soft penalty formulation, however, is almost universally applied to constrained (i.e. physics informed) ML problems, and whether the hard constraint method holds advantages in the sort of problems with genuinely conflicting constraints, such as those, for example, considered by [36, 37], or involve training using observed (i.e. noisy) data is an area for future work.

The other notable outcome of this work has been the recognition that some equations—ODE-4, ODE-6, ODE-7 and ODE-8—admit trivial (i.e. $y = 0$) solutions, and that sometimes GP preferentially discovers these by solving the literal collocation problem rather than the required ODE. As far as we aware, this observation has only previously been reported in [10] in a symbolic regression setting. That a multiobjective GP, which prefers simpler solutions given the same loss, should opportunistically discover trivial solutions is maybe understandable, but the exact mechanism is unclear. We have conjectured [10] that this subset of problematic ODEs have some sort of 'basin of attraction' in their loss functions, and demonstrated that adding a threshold on the 'energy' in the evolving solutions can, in some cases. steer the GP away from trivial solutions. This extra threshold constraint was never necessary for ODE-6 whereas ODE-7 proved the most challenging equation. Whether the numerical optimizations associated with the different ODEs present different levels of non-convexity [7] could be one cause although in [10] we noted that trivial solutions are generated even without constant tuning suggesting that the problem is fundamentally associated with the the loss function [25].

For convenience, Table 23 consolidates the results from Tables 6, 11 and 16 from which we can observe that the hard-constraint method is consistently the most effective in suppressing trivial solutions for the problematic ODEs, while the penalty method is the least effective, notably for ODE-7. This is unsurprising as the penalty method imposes the threshold constraint as a *soft* (i.e. optional) constraint as opposed to its hard imposition in the **hc** and **dp** methods. The penalty method may



(a) ODE-1, 10 collocation points

(b) ODE-2, 10 collocation points

(c) ODE-3, 100 collocation points

(d) ODE-4_t, 100 collocation points

(e) ODE-5, 100 collocation points

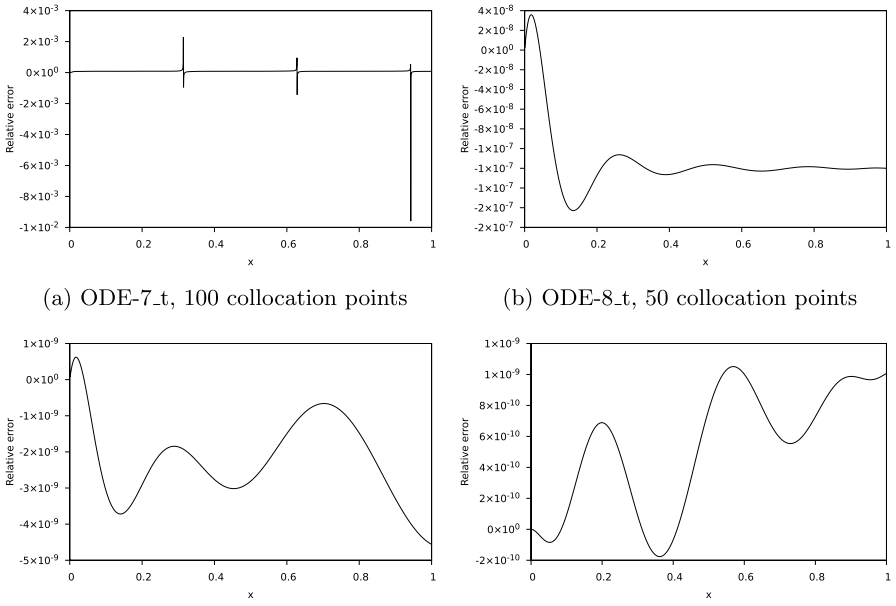
(f) ODE-6, 10 collocation points

Fig. 2 Best hard-constraint (**hc**) results for the first-order ODEs

thus not be able to impose the threshold constraint in circumstances where there is a conflict between trivial and desired solutions.

Returning to Table 21, quite why the penalty method should be so consistently poor for ODE-1 and ODE-2 is again unclear. This also needs to be considered alongside the fact that ODE-6 proved no problem in terms of producing trivial solutions, ODE-4, ODE-7 and ODE-8 required the use of a threshold constraint, but ODE-7 proved the most challenging. The clear implication is that some property of the equations is involved, and identifying this is future work.

Turning to RQ3, the research question that asked whether the restricted set of GP functions is able to synthesize general solutions, the answer would appear to be an emphatic 'yes'. We repeat that none of the known solutions of the benchmark ODEs can be exactly represented with our function set, yet we are able to produce accurate



(a) ODE-7_t, 100 collocation points

(b) ODE-8_t, 50 collocation points

(c) ODE-9, 100 collocation points

(d) ODE-10, 100 collocation points

Fig. 3 Best hard-constraint (**hc**) results for the second-order ODEs

Table 22 Maximum absolute relative errors from fitting 4th-, 6th- and 8th-order polynomials to the ODEs' analytical solutions

Order	ODE-1	ODE-2	ODE-3	ODE-4	ODE-5	ODE-6
4	0.159	0.134	1.62×10^{-3}	1.93×10^{-4}	9.97×10^{-3}	5.89×10^{-4}
6	4.16×10^{-2}	3.50×10^{-2}	4.67×10^{-6}	1.00×10^{-6}	6.95×10^{-4}	1.63×10^{-5}
8	9.39×10^{-3}	7.90×10^{-3}	6.56×10^{-9}	1.79×10^{-8}	6.16×10^{-5}	3.95×10^{-7}

	ODE-7	ODE-8	ODE-9	ODE-10
4	36.032	29.638	5.68×10^{-2}	3.76×10^{-3}
6	10.579	0.448	6.50×10^{-4}	2.53×10^{-5}
8	0.665	2.97×10^{-3}	2.17×10^{-6}	8.42×10^{-8}

solutions for all the ODEs, in some cases to within double-precision floating-point rounding error.

Since we are not aware of any comparable performance figures having been published in the literature, comparing our relative error performance values with previous work is not possible. Typically, researchers have only plotted their approximation overlaid with the known, true solution, and noted that the errors are not discernible (at the scales of the plots). Raissi et al. [4] quote figures for *relative* L_2 *norm* for PDE solutions using DNNs in the range 10^{-4} to 10^{-5}. Recently, Hao et al. [38] have published a PDE benchmark for physics-informed ML in which they also proposed relative L_2 as one convenient summary statistic, and cite values of 10^{-2} to 10^{-4} as best performance figures, albeit again for PDEs. Table 24 shows the

Table 23 Consolation of Tables 6, 11 and 16 showing the numbers of non-trivial solutions from the 30 repetitions for ODE-4, ODE-7 and ODE-8 with and without the additional threshold constraint for varying numbers of collocation points

# Collocation pts.	Penalty method				Death penalty				Hard constraint			
	10	20	50	100	10	20	50	100	10	20	50	100
ODE-4 without	10	19	25	23	0	6	5	8	8	11	22	21
ODE-4 with	30	30	30	30	30	30	30	30	30	30	30	30
ODE-7 without	0	0	0	0	0	0	0	0	0	0	0	0
ODE-7 with	0	1	1	6	30	30	30	30	30	30	30	30
ODE-8 without	0	0	4	9	0	0	0	0	0	0	0	0
ODE-8 with	30	30	30	30	24	26	26	26	30	30	30	30

Table 16 Numbers of non-trivial solutions from the 30 repetitions for ODE-4, ODE-7 and ODE-8 with and without the additional threshold constraint for varying numbers of collocation points for hard constraint method

ODE	# Collocation pts.			
	10	20	50	100
ODE-4 without	8	11	22	21
ODE-4 with	30	30	30	30
ODE-7 without	0	0	0	0
ODE-7 with	30	30	30	30
ODE-8 without	0	0	0	0
ODE-8 with	30	30	30	30

Table 17 Comparison of the performance of the three methods for ODE-1 and 50 collocation points

			Method			
			Death penalty		Penalty method	
Method	Hard	Constr.	0.070	0.104	1.0	**1.0**
			− 2.49e−31	− 1.78e−15	− 2.41e−8	**− 1.17e−4**
			4.35e−19	1.00e−12	− 4.71e−9	**− 3.45e−5**
			4.21e−15	5.77e−11	− 5.97e−11	**− 1.21e−5**
	Death	Penalty			1.0	**1.0**
					− 2.26e−8	**− 1.26e−4**
					− 4.75e−9	**− 3.45e−5**
					− 5.97e−11	**− 1.28e−5**

See Sect. 3.1.1 for an explanation of the table layout

Table 18 Comparison of the performance of the three methods for ODE-2 and 50 collocation points

			Method			
			Death penalty		Penalty method	
Method	Hard	Constr.	0.00906	0.62504	**1.0**	**1.0**
			− 1.25e−9	− 4.87e−9	**− 2.14e−5**	**− 3.39e−3**
			1.01e−8	− 4.31e−12	**− 8.79e−6**	**− 1.93e−3**
			3.69e−8	8.82e−10	**− 2.30e−6**	**− 9.05e−4**
	Death	Penalty			**1.0**	**1.0**
					− 2.14e−5	**− 3.38e−3**
					− 9.80e−6	**− 1.93e−3**
					− 2.32e−6	**− 9.05e−4**

See Sect. 3.1.1 for an explanation of the table layout

relative L_2 values for the best-of-run results shown in Figs. 2 and 3 from which it is clear that the relative L_2 values here are in the approximate range 10^{-8} to 10^{-32}. Although not directly comparable with the values in Hao et al., we can infer that the accuracies of the best-of-run solutions are high. These figures also act a benchmark for future work.

As pointed out above, the present setting is 'unusual' from a machine learning standpoint in that we are solving an equation, not inducing a function from a noisy dataset. We can thus *directly* use the training loss in the model selection phase—we have no need to consider overfitting to a particular data sample. The corollary of this is that for practical situations where the analytical solution to the ODE is not known, we can use the training loss a measure of solution quality: it follows that a

Table 19 Comparison of the performance of the three methods for ODE-5 and 50 collocation points

			Method			
			Death penalty		Penalty method	
Method	Hard	Constr.	0.0	0.968	0.793	1.0
			4.71e−13	− 2.25e−11	− 1.22e−12	− 8.25e−09
			6.99e−13	− 7.19e−12	− 1.24e−13	− 3.57e−09
			1.16e−12	− 3.08e−13	3.73e−13	− 2.10e−09
	Death	Penalty			1.0	1.0
					− 2.07e−12	− 8.60e−09
					− 1.39e−12	− 3.99e−09
					− 7.72e−13	− 2.27e−09

See Sect. 3.1.1 for an explanation of the table layout

Table 20 Comparison of the performance of the three methods for ODE-7 with threshold and 50 collocation points

			Method					
			Death penalty			Penalty method		
Method	Hard	Constr.	**0.999**	0.984	0.970	**1.000**	0.002	0.184
			− 5.98e−02	− 1.29e−07	− 2.44e−15	**− 4.77e−01**	0.00e+00	0.00e+00
			− 4.82e−02	− 8.62e−10	− 1.03e−19	**− 4.77e−01**	0.00e+00	0.00e+00
			− 3.77e−02	− 3.55e−15	0.00e+00	**− 4.77e−01**	2.09e−11	6.04e−21
	Death	Penalty				**1.0**	0.0	0.001
						− 4.37e−01	2.84e−14	0.00e+00
						− 4.29e−01	7.14e−09	1.09e−19
						− 4.14e−01	1.26e−07	2.44e−15

See Sect. 3.1.1 for an explanation of the table layout

small training loss at all points in the domain implies that the ODE is being well satisfied. Figure 4 shows both the relative error and the training loss over the domain for two examples of ODE-9. Figure 4b shows an obviously smaller training loss than Fig. 4a. It is clear from these two figures that a better training loss implies a better solution. An area of future work is to propagate the training loss values through to errors on the actual evolved solution thereby providing an error bound when the ODE's solution is not known a priori.

It is also instructive to examine the self-diagnostic information available for ODE-7 since this proved a difficult case; Fig. 5 shows the same result as Fig. 3a. Here the largest training loss occurs near $x \approx 0$ and thereafter takes on much smaller values. This figure reinforces the point that the spikes in the previously presented relative error plot are a numerical issue with calculating the relative error,

Important to the interpretation of the data in this paper has been our consideration of effect sizes. In many cases, the statistical test returns a result that suggests a high level of significance, but the spread of actual difference values is small, in some cases extremely small. This is well-known in the statistics literature—see, for example, [32]—and can be illustrated with the case of a set of thirty differences, each of $+1 \times 10^{-100}$. A statistical test on these data, frequentist or Bayesian, would return very strong evidence of difference despite the obvious conclusion that the performance of the two methods being compared differs only their 100-th decimal places, something that is highly unlikely to

Table 21 Summary of comparison of methods for all benchmark ODEs

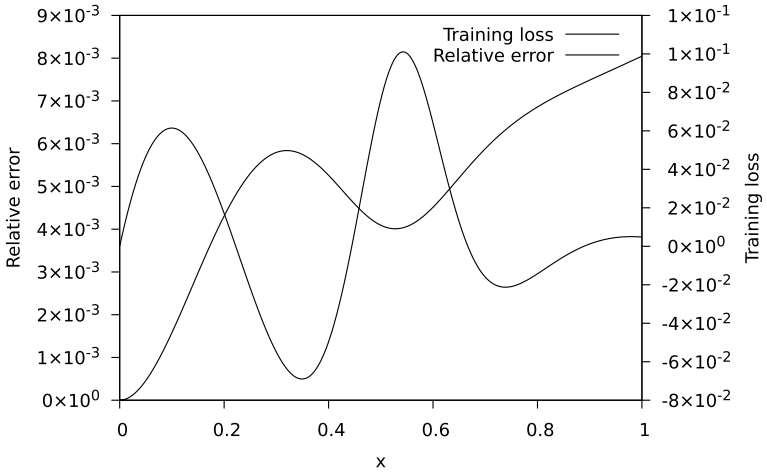
		Test error	IVV-1	IVV-2
ODE-1	10		**hc≺pm**, **dp≺pm**	–
	20		**hc≺pm**, **dp≺pm**	–
	50		**hc≺pm**, **dp≺pm**	–
	100		**hc≺pm**, **dp≺pm**	–
ODE-2	10	**hc≺pm**, **dp≺pm**	**hc≺pm**, **dp≺pm**	–
	20	**hc≺pm**, **dp≺pm**	**hc≺pm**, **dp≺pm**	–
	50	**hc≺pm**, **dp≺pm**	**hc≺pm**, **dp≺pm**	–
	100	**hc≺pm**, **dp≺pm**	**hc≺pm**, **dp≺pm**	–
ODE-3	10			-
	20			–
	50			–
	100			–
ODE-4_t	10	**hc≺dp**, **pm≺dp**		–
	20	**hc≺dp**, **pm≺dp**		–
	50	**hc≺dp**, **pm≺dp**		–
	100	**hc≺dp**, **pm≺dp**		–
ODE-5	10			–
	20			–
	50			–
	100			–
ODE-6	10			–
	20			–
	50			–
	100			–
ODE-7_t	10	**hc≺pm**, **dp≺pm**		
	20	**hc≺dp≺pm**		
	50	**hc≺dp≺pm**		
	100	**hc≺dp≺pm**		
ODE-8_t	10	**hc≺dp**, **pm≺dp**	**hc≺dp**	
	20	**hc≺dp**, **pm≺dp**		
	50	**hc≺dp**, **pm≺dp**	**hc≺dp**	
	100	**hc≺dp**, **pm≺dp**		
ODE-9	10			
	20			
	50			
	100			
ODE-10	10			
	20			
	50			
	100			

'IVV-1' = the initial value violation, and 'IVV-2' = violation of the initial derivative value (for 2nd-order equations only.) See text for further details

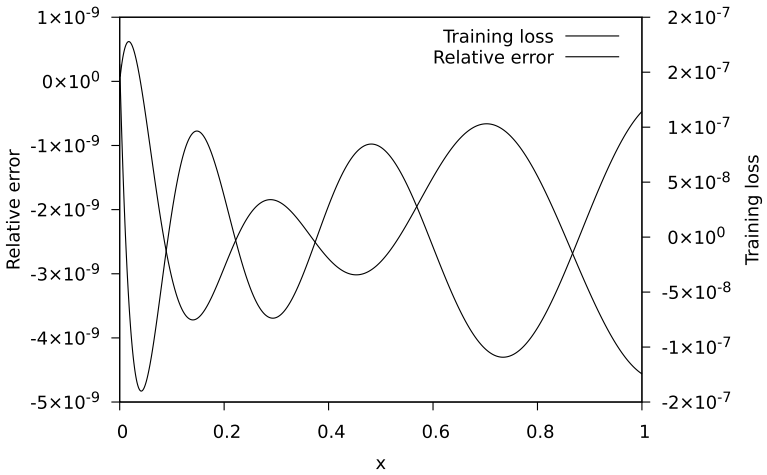
be of any practical interest. Had we only reported the outcomes of the statistical tests in this paper, and omitted consideration of effect sizes, the conclusions would likely have been lacking any coherent pattern, and our explanations convoluted and highly caveated. It is noteworthy that much of the GP literature currently reports only statistical tests leading to the obvious question of whether these results really are meaningful in terms of *effect*. To quote the much-cited paper by Ioannidis [39]:

Table 24 Relative L_2 values for the best-of-run results shown in Figs. 2 and 3

ODE-1	ODE-2	ODE-3	ODE-4t	ODE-5	ODE-6
1.1×10^{-32}	1.3×10^{-13}	1.0×10^{-21}	5.7×10^{-17}	5.1×10^{-15}	1.1×10^{-31}
ODE-7t	ODE-8t	ODE-9	ODE-10		
7.2×10^{-9}	1.4×10^{-13}	7.9×10^{-18}	7.2×10^{-19}		



(a) First instance of ODE-9.



(b) Second instance of ODE-9.

Fig. 4 Relative error (left ordinate scale) and training loss (right ordinate scale) over the domain for two examples of ODE-9 with hard constraints

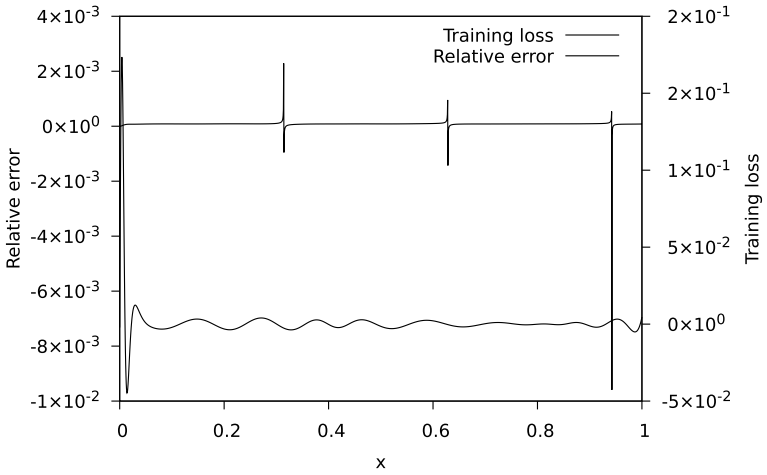


Fig. 5 Relative error (left ordinate scale) and training loss (right ordinate scale) over the domain for ODE-7 with threshold/hard constraints with 100 collocation points

> "…the high rate of nonreplication (lack of confirmation) of research discoveries is a consequence of the convenient, yet ill-founded strategy of claiming conclusive research findings solely on the basis of a single study assessed by formal statistical significance, typically for a p-value less than 0.05."

The clear suggestion to the GP community, therefore, is to include consideration of effect sizes in their analyses.

Extension to partial differential equations (PDEs) is an obvious area of future work although PDEs are a more complicated arena. For example, a given PDE can have multiple as well as weak solutions [40] and so will require careful study. The present approach can be easily extended to, say, finding a solution $y(x_1, x_2)$ to a 2D PDE by minimizing a loss function like Eq. (5) calculated over a two-dimensional grid of collocation points in x_1 and x_2.

5 Conclusions

In this paper we have reported the solution of a set of benchmark ordinary differential equations (ODEs) using multiobjective genetic programming with tuning of the tree parameters (constants) at each iteration. We compare the results of three evolutionary methods: i) the conventional soft penalty method whereby constraints are embedded as weighted terms in the loss function (**pm**), ii) a method where the initial value constraints are imposed as hard constraints in the constant optimization algorithm, but if any tree is unable to meet any of the constraints after optimization, it is given an 'infinite' fitness, the so-called *death penalty* (**dp**), and iii) a method that is similar to the previous method except any tree that cannot meet the constraints is ordered using a novel constraint handling method (**hc**) such that violations are

sorted using Pareto ranking, and that two infeasible trees are ordered according to the extent of their relative constraint violations..

In terms of investigating the numbers of collocation points necessary, results for a few of the ODEs suggest that 10, or in some cases 20, points is too few, but that 50 collocation points is sufficient.

As to whether any of the three considered methods is superior, for the majority of cases, all three methods appear statistically indistinguishable in both test error and the attainment of the initial conditions. Interestingly, the methods did not differ in meeting the initial derivative conditions, which might be considered more sensitive to approximation than the initial function value condition. We conjecture that this may be the consequence of using parameter tuning. Notwithstanding, for two of the benchmark ODEs the **pm** method appears to perform especially poorly.

Where performance differences did emerge, the **hc** method using the novel constraint handling mechanism was always the best or equal best suggesting that this should be the method of choice although we also infer that for many cases, the soft penalty formulation may be sufficient because the minimization of the goodness-of-fit term and constraint violation term in the loss are not in conflict. This judgment might well change in settings where the loss contains conflicting terms.

The other notable observation from this work was the generation of trivial (i.e. $y = 0$) solutions, which we suspect is due to the evolutionary algorithm in some cases solving the *literal* collocation problem rather than the ODE itself. Although we have demonstrated a partial suppressive measure—adding a further constraint to ensure that the 'energy' content of the solution exceeded a positive threshold—this is not foolproof. Moreover, one of the benchmark ODEs that could, in principle, have produced trivial solutions never did so. Other test ODEs, however, proved challenging in terms of being able to produce non-trivial solutions. We infer that the solution method cannot be separated from the properties of the ODEs, and that this requires further theoretical insight into the ODEs being solved.

In practice, the best generated solutions are of greatest interest—we consider those with the smallest training loss. For each of the benchmark ODEs, we have generated solutions with (very) small relative errors, and in some cases, relative errors that are within floating-point rounding error of the known solutions. It is noteworthy that none of the benchmark ODEs can be constructed exactly with the GP function set used in this work suggesting that the approximation abilities of our GP framework are sufficient without the need to include transcendental or other functions.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10710-026-09536-x.

Author contributions This is a single author paper.

Data availability Provided as supplementary material

Declarations

Conflict of interests The authors declare no conflict of interests.

Ethical approval This work did not require ethics approval.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. D.W. Jordan, P. Smith, *Nonlinear ordinary differential equations: An introduction for scientists and engineers*, 4th edn. (Oxford University Press, Oxford, UK, 2007)
2. I.G. Tsoulos, I.E. Lagaris, Solving differential equations with genetic programming. Genet. Program. Evol. M. **7**, 33–54 (2006)
3. T. Seaton, G. Brown, J. F. EMiller, A. I. sparcia-Alcázar, A. Ekárt, S. Silva, S. Dignum, A. S. Etaner-Uyar, (eds) *Analytic solutions to differential equations under graph-based genetic programming*. (eds A. I. Esparcia-Alcázar, A. Ekárt, S. Silva, S. Dignum, A. S.Etaner-Uyar,) *13th European Conference on Genetic Programming (EuroGP'10)*, 232–243 (Istanbul, Turkey, 2010)
4. M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
5. L.B. Rall, The arithmetic of differentiation. Math. Mag. **59**, 275–282 (1986)
6. S. Cuomo et al., Scientific machine learning through physics-informed neural networks: where we are and what's next. J. Sci. Comput. **92**, 88 (2022)
7. J. Ramirez, M. Hashemizadeh, S. Lacoste-Julien, Position: adopt constraints over penalties in deep learning. *CoRR* arXiv:2505.20628 (2025). https://doi.org/10.48550/arXiv.2505.20628
8. W. J. A. Lobão, D. M. Dias, M. A. C. Pacheco, D. E. Goldberg, C. A. C. Coello, X. Zhang, (eds) *Genetic programming and automatic differentiation algorithms applied to the solution of ordinary and partial differential equations*. (eds D. E. Goldberg, C. A. C. Coello, X.Zhang,) *IEEE Congress on Evolutionary Computation*, 5286–5292 (Vancouver, Canada, 2016)
9. H. Oh, et al., Genetic programming based symbolic regression for analytical solutions to differential equations (2023). arXiv:2302.03175
10. P. Rockett, The solution of ordinary differential equations using genetic programming with constant tuning. 24th UK Workshop on Computational Intelligence, Edinburgh (2025)
11. P. Rockett, Constant optimization and feature standardization in multiobjective genetic programming. Genet. Program. Evol. M. **23**, 37–69 (2021)
12. J. Nocedal, S.J. Wright, *Numerical optimization*, 2nd edn. (Springer, New York, 2006)
13. L. Lu et al., Physics-informed neural networks with hard constraints for inverse design. SIAM J. Sci. Comput. **43**, B1105–B1132 (2021)
14. P. Márquez-Neila, M. Salzmann, P. Fua, Imposing hard constraints on deep networks: promises and limitations. CoRR (2017). arXiv:1706.02025
15. J. Ni, R.H. Drieberg, P.I. Rockett, The use of an analytic quotient operator in genetic programming. IEEE Trans. Evol. Comput. **17**, 146–152 (2013)
16. J.J. Moré, *The Levenberg-Marquardt algorithm: Implementation and theory* (Springer, Berlin, Heidelberg, 1978), pp.105–116
17. P. Rockett, Y. Kaszubowski Lopes, T. Dou, E. A. Hathway, H. P. García, L. Sánchez-González, M. C. Limas, H. Quintián-Pardo, E. S. C. Rodríguez, (eds) *d(Tree)-by-dx: Automatic and exact differentiation of genetic programming trees*. (eds H. P. García, L. Sánchez-González, M. C. H. Limas, Quintián-Pardo, E. S. C. Rodríguez,) 14th International Conference on Hybrid Artificial Intelligent Systems (HAIS2019), 133–144 (León, Spain, 2019)

18. C.C. Margossian, A review of automatic differentiation and its efficient implementation. WIREs Data Min. Knowl. Discov. **9**, e1305 (2019)
19. I. Rahimi, A.H. Gandomi, F. Chen, E. Mezura-Montes, A review on constraint handling techniques for population-based algorithms: From single-objective to multi-objective optimization. Arch. Comput. Method. E. **30**, 2181–2209 (2023)
20. T. Dou, P. Rockett, Comparison of semantic-based local search methods for multiobjective genetic programming. Genet. Program. Evol. M. **19**, 535–563 (2018)
21. C. Fonseca, P.J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: a unified formulation. IEEE Trans. Syst. Man Cybern. Part A Syst. Humans **28**, 26–37 (1998)
22. R. Poli, W. B. Langdon, N. F. McPhee, *A Field Guide to Genetic Programming* (Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008)
23. W. Rudin, *Principles of Mathematical Analysis* (McGraw-Hill, 1976)
24. E. Mezura-Montes, C. A. Coello Coello, E. I. Tun-Morales, R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, H. Sossa, (eds) *Simple feasibility rules and differential evolution for constrained optimization*. (eds R. Monroy, G. Arroyo-Figueroa, L. E. H. Sucar, Sossa,) *MICAI 2004: Advances in Artificial Intelligence*, 707–716 (Mexico City, Mexico, 2004)
25. F. M. Rohrhofer, S. Posch, C. Gößnitzer, B. C. Geiger, On the role of fixed points of dynamical systems in training physics-informed neural networks. Trans. Mach. Learn. Res. **2023** (2023)
26. M. Penwarden, A.D. Jagtap, S. Zhe, G.E. Karniadakis, R.M. Kirby, A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions. J. Comput. Phys. **493**, 112464 (2023)
27. J.K. Kruschke, Bayesian analysis reporting guidelines. Nat. Hum. Behav. **5**, 1282–1291 (2021)
28. A. Benavoli, G. Corani, F. Mangili, M. Zaffalon, F. Ruggeri, E. P. Xing, T. Jebara, (eds) *A Bayesian Wilcoxon signed-rank test based on the Dirichlet process*. (eds E. P. Xing, T. Jebara,) 31st International Conference on Machine Learning, Vol. 32, 1026–1034 (Bejing, China, 2014)
29. A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. J. Mach. Learn. Res. **18**, 1–36 (2017)
30. S. Greenland et al., Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations. Eur. J. Epidemiol. **31**, 337–350 (2016)
31. R.E. Kass, A.E. Raftery, Bayes factors. J. Amer. Stat. Assoc. **90**, 773–795 (1995)
32. J.K. Kruschke, T.M. Liddell, The Bayesian new statistics: Hypothesis testing, estimation, meta-analysis, and power analysis from a Bayesian perspective. Psychon B Rev **25**, 178–206 (2018)
33. J. Cohen, Things I have learned (so far). Am. Psychol. **45**, 1304–1312 (1990)
34. B. Efron, Better bootstrap confidence intervals. J. Amer. Stat. Assoc. **82**, 171–185 (1987)
35. T. Dou, Y. Kaszubowski Lopes, P. Rockett, E.A. Hathway, E. Saber, GPML: an XML-based standard for the interchange of genetic programming trees. Genet. Program. Evol. M. **21**, 605–627 (2020)
36. J. Kubalík, E. Derner, R. Babuška, C. A. Coello Coello, (ed.) *Symbolic regression driven by training data and prior knowledge*. (ed.Coello Coello, C. A.) *Genetic and Evolutionary Computation Conference (GECCO '20)*, 958–966 (Cancún Mexico, 2020)
37. C. Haider, F. Bachinger, F. Holzinger, F. O. de França, A. Quesada-Arencibia, M. Affenzeller, R. Moreno-Díaz, (eds) *Comparing constraint evaluation methods for shape-constrained regression*. (eds Quesada-Arencibia, A., Affenzeller, M. & Moreno-Díaz, R.) 19th International Conference on Computer Aided Systems Theory (EUROCAST 2024), 68–76 (Las Palmas de Gran Canaria, Spain, 2024)
38. Z. Hao, et al. A. Globerson, et al. (eds) *PINNacle: A comprehensive benchmark of physics-informed neural networks for solving PDEs*. (eds Globerson, A. et al.) *Advances in Neural Information Processing Systems*, Vol. 37, 76721–76774 (2024)
39. J.P.A. Ioannidis, Why most published research findings are false. PLoS Med. **2**, e124 (2005)
40. L.C. Evans, *Partial differential equations*, 2nd edn. (American Mathematical Society, Providence, RI, 2010)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.