

Genetic Programming Based Symbolic Regression for Analytical Solutions to Differential Equations

Hongsup Oh^a, Roman Amici^{b,c}, Geoffrey Bomarito^d, Shandian Zhe^c, Robert M. Kirby^{b,c}, Jacob Hochhalter^a

^a*Department of Mechanical Engineering, University of Utah*

^b*Scientific Computing and Imaging Institute, University of Utah*

^c*School of Computing, University of Utah*

^d*NASA Langley Research Center*

Abstract

In this paper, we present a machine learning method for the discovery of analytic solutions to differential equations. The method utilizes an inherently interpretable algorithm, genetic programming based symbolic regression. Unlike conventional accuracy measures in machine learning we demonstrate the ability to recover true analytic solutions, as opposed to a numerical approximation. The method is verified by assessing its ability to recover known analytic solutions for two separate differential equations. The developed method is compared to a conventional, purely data-driven genetic programming based symbolic regression algorithm. The reliability of successful evolution of the true solution, or an algebraic equivalent, is demonstrated.

Keywords: Physics-informed machine learning, Symbolic regression, Genetic programming, Boundary-value problems

1. Introduction

The governing physics in engineering mechanics problems is often formalized mathematically via ordinary or partial differential equations (ODEs or PDEs, respectively). However, direct analytical solutions of these equations are generally not attainable in practice other than in idealized cases. For practical engineering problems involving challenges such as complex geometries, discretization-based methods, *e.g.*, finite element (FE) analysis, are utilized to provide numerical approximations of the solution. In recent years, data-driven machine learning (ML) alternatives have become widespread, and methods have been developed to incorporate knowledge

of domain physics: often dubbed theory-guided, physics-informed, or physics-regularized ML Karniadakis et al. (2021); Raissi et al. (2019); Karpatne et al. (2017). Inspired by the work of Raissi et al. (2017), wherein physics-informed neural networks (PINNs) were demonstrated for finding approximate solutions of PDEs, we demonstrate physics-regularization within an inherently interpretable ML method for determination of symbolic solutions to ODEs and PDEs. While the developed method seeks minimization of a residual for approximate symbolic solutions it is demonstrated that true analytical solutions can be reliably discovered.

In practice, data in engineering programs are typically expensive to acquire, especially at a scale sufficient to train most conventional ML models. This often forces decisions based on relatively small datasets. However, recent advances in high-throughput data acquisition methods to support ML are promising Heckman et al. (2020). Complementing data acquisition in the ML process are

Email addresses: hongsup.oh@utah.edu (Hongsup Oh), amicir@gmail.com (Roman Amici), geoffrey.f.bomarito@nasa.gov (Geoffrey Bomarito), zhe@cs.utah.edu (Shandian Zhe), kirby@cs.utah.edu (Robert M. Kirby), jacob.hochhalter@utah.edu (Jacob Hochhalter)

physics-regularized approaches which can promote generalizable models that are consistent with *a priori* knowledge Raissi et al. (2017, 2019); Warner et al. (2020). Although the area of PINNs is especially promising, black-box ML methods applied to scientific and engineering applications face challenges related to the interpretability and explainability of the models. Unfortunately, opening the metaphorical ML black box does not provide immediate insight into how or why a model works: a necessary step in engineering and science scenarios. To remedy this opacity, the current trend is to utilize additional tools (*e.g.*, visualization, weight-matrix analysis, or decision trees) to aid interpretation of the resulting complex solutions Adadi and Berrada (2018); Došilović et al. (2018). The knowledge-extraction process is then analogous to current discretization-based paradigms (*e.g.*, FE): evaluate the approximate solution at a set of points and generate a contour plot or data table. In the context of engineering mechanics, lacking interpretability and explainability can perpetuate an inherent lack of understanding about the generated model and its prediction capabilities and limitations. Furthermore, in many cases, there is little or no added accuracy benefit to using such black-box models compared to interpretable alternatives Rudin and Radin (2019).

Herein, genetic programming based symbolic regression (GPSR) is used due to its inherent interpretability and recent successful application to physics and mechanics problems Versino et al. (2017); Bomarito et al. (2021); Hernandez et al. (2019). The standard form of GPSR has been shown to be capable of developing true, verifiable models in the area of solid mechanics Bomarito et al. (2021) and of discovering underlying physics from data Schmidt and Lipson (2009). Extending beyond previous works, we augment standard GPSR with a physics-regularized fitness function, PR-GPSR, to evolve solutions to known differential equations *i.e.*, where the governing mechanics are known and a solution is sought. GPSR produces models in the form of analytic equations Koza and Koza (1992); thus, when regularized by the residual of the known ODE or PDE, it aims to produce analytic solutions thereof. As will be demonstrated herein for symbolic models, as is also true of PINNs, physics-regularized ML requires only (at a minimum) a complete statement of the differential equation and boundary or initial conditions necessary for a well-posed problem.

Due to the free-form symbolic regression nature of GPSR, the application of physics-regularization becomes analogous to the determination of closed-form solutions to differential equations and as such distinguished from more conventional numerical approximations. For example, in a conventional mathematical approach the practitioner proposes an ansatz space from which possible analytic solutions can be built, as well as the allowable ways in which expressions can be assembled (*e.g.*, addition, multiplication, composition, *etc.*). In the language of ML, the building blocks or dictionaries of mathematical functions are assembled and evolved to evaluate possible analytical solutions. GPSR with a physics-regularized fitness function essentially automates this process of practitioner proposition of analytic solutions. In the context of engineering mechanics, a beneficial output of this approach is that symbolic models fit naturally within existing workflows. When coupled with user interpretation, and explainability via satisfaction of known differential equations, this promotes increased trust, accessibility, and transfer of generated ML models into practice. Further, more insights readily gained from symbolic equations can suggest future data acquisition or new theories by identifying model characteristics *e.g.*, asymptotes. Lastly, symbolic equations allow for a broader mathematical treatment of produced models *e.g.*, formulation of analytic adjoints.

The paper is organized as follows. In Section 2, we present a brief overview of GPSR and provide details on how we apply GPSR for learning the solutions to differential equations. In Section 3 we define two numerical experiments based on boundary-value problems. In Section 4, we demonstrate the performance of the developed method. In Section 5, we present a discussion of results with a summary of the successes and challenges of the developed methodology. We conclude in Section 6 with a summary and a vision for next steps.

2. Methods

2.1. Genetic Programming for Symbolic Regression (GPSR)

Symbolic regression is a method that aims to model an input data set without assuming its form. Instead, candidate models are proposed and evaluated by the algorithm,

with the only assumption being that the data can be modeled by some algebraic expression. This approach is in contrast to traditional regression methods in which model form selection is made first, and the regression method then estimates the model parameters. SR reformulates the traditional regression problem into that of searching for an optimal model form and its associated parameters. Note that this is similar to what is done in deep neural networks (DNNs), in which through a non-linear composition of layers a regressor learns both coefficients and basis functions (model form) from data. However, in the case of SR, the fundamental mathematical building blocks from which the model form is constructed are defined by the user.

From a general perspective, SR is an optimization problem that occurs over a non-numeric domain of mathematical operators. There are binary operators which utilize two operands (*e.g.*, +, −, ×, ÷) and functions (unary operators) which utilize one operand (*e.g.*, sin, cos, exp, ln). An SR model is then characterized by a variable-length combination of these operators and coefficients and, therefore, poses an infinite space of possible model forms to search. In practice, the mathematical operator domain is limited by a finite set of operations and maximum model complexity threshold. While these practical constraints help constrain the vast search space an efficient search method is required to discover accurate models.

To search the space, genetic programming (GP) is the most commonly used approach to SR, together termed GPSR. Within GPSR, genetic algorithms are used to evolve models based on their fitness relative to a specified fitness function(s), which are discussed in the next section. This fitness function is used to select models most likely to perform better, after which model evolution occurs through random recombination (*i.e.*, crossover) and permutation (*i.e.*, mutation) to generate new candidate models. At the same time, the candidates with poorest fitness are dropped out of the population (*e.g.*, natural selection). The iterative exploration of the solution space is subject to both randomization and guidance from the particular fitness and crossover and mutation procedures implemented.

Potential GPSR fitness functions include standard explicit error metrics (*e.g.*, mean squared error), custom reward-cost functions, or derivative-based fitness func-

tions for implicit equations Schmidt and Lipson (2010); Bomarito et al. (2021). Combining these with alternatives for solution representation/structure (acyclic graphs) and evolution strategies make GPSR suitable to a variety of applications. However, the expense for this flexibility and benefits includes increased computational resources, potential non-determinacy, and susceptibility to selecting high-variance solutions Wang et al. (2019).

Here, we use the open-source NASA GPSR code Bingo NASA (2013); Randall et al. (2022) because of its modular nature, allowing for implementation of custom fitness functions, and its ability to scale to high performance computing (HPC) resources. To improve the performance and robustness within Bingo, several efforts are made, such as deterministic crowding Mahfoud (1995) and parallel island evolution Fernandez et al. (2005); Whitley et al. (1999). Deterministic crowding is a common niching algorithm to avoid converging to local optima wherein pairs of models are generated based on their similarities, and best fit individuals survive to the next generation. Further, we implement a parallel island evolution strategy whereby multiple islands (archipelago) are distributed across available computer cores. Individual islands then independently evolve with periodic communication of models among the islands to help maintain diversity.

2.2. Physics-regularized fitness function

Setup begins with a set of training data (X_i, y_i) where $i \in 1, 2, \dots, n$ and X_i is a p -dimensional vector-valued input of features, $X_i = x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(p)}$, and y_i are the corresponding labels. Hereafter, the index on X is dropped for brevity as a feature vector is always implied. A model, $f : \mathbb{R}^p \rightarrow \mathbb{R}$, is then sought for these training data. For each model proposed by GPSR, $\tilde{f}(X)$, a defined fitness function quantifies its accuracy and is sought to be minimized. Conventionally, a purely data-driven fitness, F^{dd} , would be defined as a vector:

$$F_i^{dd} = \tilde{f}(X) - y_i \quad (1)$$

or homogenized as, for example, a mean-squared error:

$$F^{dd} = \frac{1}{n} \sum_{i=1}^n (F_i^{dd})^2. \quad (2)$$

To impose physics regularization, the fitness is augmented with a measure of how well \tilde{f} satisfies the prescribed differential equations, $L^{(k)}(\tilde{f}(X^{(k)}))$, where L is an arbitrary differential operator. Here, $k \in 1, 2, \dots, l$, represents l differential equations *e.g.*, various boundary conditions or multiple governing physics. Further, the $X^{(k)}$ at which these differential equations are evaluated need not be coincident with the purely data-driven inputs, X , nor consistent across the $L^{(k)}$. A purely physics-regularized fitness, F^{pr} , can then be defined as:

$$F_j^{pr} = [\lambda^{(1)}L^{(1)}(\tilde{f}(X^{(1)})); \lambda^{(2)}L^{(2)}(\tilde{f}(X^{(2)})); \dots \lambda^{(l)}L^{(l)}(\tilde{f}(X^{(l)}))] \quad (3)$$

where the semi-colon indicates concatenation into a one dimensional vector and $j \in 1, 2, \dots, m$, where m is the total number of $X^{(k)}$ data points. Similar to Equation 2, this vector-valued fitness can then be homogenized as, for example, a mean-squared error. This second, physics-regularized, fitness term includes optional hyperparameters, $\lambda^{(k)}$, which control a relative weighting among the conventional training data and the k differential equations. It is often sufficient to set all $\lambda^{(k)} = 1$, as is done herein, and results in a complete, concatenated, vector-valued fitness:

$$F_{n+m} = [F_i^{dd}; F_j^{pr}], \quad (4)$$

or, can also be homogenized as, for example, a mean-squared error which is referred to hereafter as F .

To evaluate the physics-regularized fitness term, F^{pr} , $L^{(k)}(\tilde{f}(X_j))$ must be computed. As described in the previous section, Bingo produces a set of mathematical operators that compose each function. To enable the computation of the requisite derivatives, we simply translate these mathematical operators into primitives provided by PyTorch¹ Paszke et al. (2019) and employ the automatic differentiation method, autograd Kokhlikyan et al. (2019). Automatic differentiation is a mature technology that has gained further prominence in recent years due to its use in training neural networks (NN). However, the NN use-case

and our own differ in an important way. NNs construct a single, highly-parameterized trial function. The resulting gradients are then correspondingly high dimensional, resulting in a higher computational demand.

Automatic differentiation frameworks tend to be designed for NNs and, therefore, favor spending greater time when the function is first constructed to make repeated computation of the gradient more efficient. In our case, however, we construct a population of analytic functions for which the derivative will be computed. The dimensionality of the necessary derivatives are also much lower (*e.g.*, 3-4 dimensions for PDEs), and the derivatives will only need to be evaluated once per function construction. Thus, techniques such as extensive graph-optimization and JIT (just in time) compilation tend to harm rather than help performance in our case. PyTorch was the fastest framework which we tried; however, it is still slower than conventional function evaluation by an order of magnitude. It is possible that algorithmic differentiation can be sped up significantly via using a framework tailored specifically for this use-case.

2.3. Local optimization of coefficients

Once a model has been formed via GPSR as in Section 2.1, commonly there are undetermined constant coefficients. The physics-regularized fitness defined in the previous section is then sought to be minimized by subjecting these constant coefficients to a local optimization step. The performance of physics-regularized GPSR is found to be highly sensitive to this local optimization step. As part of this work, many of the optimization algorithms made available through the `scipy.optimize.minimize` and `scipy.optimize.root` modules in Python were tested Virtanen et al. (2001). For vector-valued fitness functions, Equation 4, root-finding methods are utilized while for scalar-valued (homogenized) fitness functions the minimization methods are utilized. Because this local optimization is automated for every GPSR model there is no model-specific customization of the initial guesses. Instead, initial guesses for the constant coefficients were selected randomly from a uniform distribution from -1 to 1. However, optimized parameters were unconstrained and, as such, could venture outside those initial guess bounds.

Because GPSR evolves arbitrary functions this assessment of available methods tests which optimization methods perform best in general. Of the available meth-

¹Specific vendor and manufacturer names are explicitly mentioned only to accurately describe the test hardware. The use of vendor and manufacturer names does not imply an endorsement by the U.S. Government nor does it imply that the specified equipment is the best available.

ods in scipy 1.8.1, the Levenberg–Marquardt (LM) algorithm performed best for vector-valued fitness functions while the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm performed best for scalar-valued fitness functions Virtanen et al. (2001). LM is a common optimization method used for non-linear least squares problems. BFGS is an iterative method used for unconstrained, non-linear optimization problems. As with many gradient-based optimization methods, both LM and BFGS can converge on local minima. Beyond simply performing best, it was found that LM and BFGS algorithms were the only two algorithms that frequently found the global optima and ran approximately 10× faster than other methods. This observation is consistent with previous findings De Melo et al. (2015). Ultimately, LM had approximately the same runtime as BFGS but more frequently determined the global optimum so it was selected for all of the numerical experiments presented next. The improved performance of LM is likely related to preservation of the vector-valued fitness function, where homogenization of fitness to a scalar value is not required.

3. Experiments

Two numerical experiments based on boundary-value problems were selected to verify that physics-regularized GPSR can evolve known solutions to differential equations. The first experiment verifies the solution to a linear, fourth-order ODE. The second experiment then verifies a linear, second-order PDE. Of particular importance with these experiments is that success occurs specifically when the known analytic solution to these differential equations evolves *i.e.*, an algebraic equivalence that is more rigorous than a conventional numerical threshold.

3.1. Euler-Bernoulli Differential Equation

The first numerical experiment is a fourth-order ODE boundary value problem, which is derived from force and moment equilibrium, and known as the Euler-Bernoulli equation. The objective of this experiment is to verify that physics-regularized GPSR can reliably evolve the known analytical solution, $u(x)$, to the known ODE:

$$L^{(1)}(u(x)) = \frac{\partial^4 u(x)}{\partial x^4} - \frac{w(x)}{EI} = 0, \quad (5)$$

where u is the deflection of the beam under applied load, w , along its length, x , E is the Young’s modulus, and I is the moment of inertia. The boundary conditions are such that the $u(x = 0) = 0$ and $u(x = l) = 0$. Similarly, the curvature, $L^{(2)} = \frac{d^2 u}{dx^2} - \kappa = 0$, is such that $\kappa(x = 0) = 0$ and $\kappa(x = l) = 0$. For this experiment, $w(x)$ is taken to be a uniform load, Figure 1, and we rewrite the constant term of Equation 5 as $\frac{w(x)}{EI} = c$.

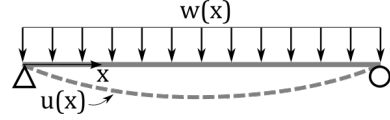


Figure 1: Simply-supported beam with uniform load.

The solution to Equation 5 can be readily obtained by integration and application of boundary conditions, upon which Equation 6 is obtained:

$$u(x) = \frac{c}{24}(x^4 - 2lx^3 + l^3x). \quad (6)$$

Representing practical values, we define $c = 5 \times 10^{-5}$ and $l = 10$, which results in the expanded form given in Equation 7. This equation consequently serves as a model form with specific coefficients that will be used to verify the produced GPSR models. Because of the symbolic nature of GPSR, we not only verify numerical accuracy, but also algebraic equivalence between the known solution and GPSR models. Note, this ability to verify model form and coefficients represents the fundamental difference between physics-regularized GPSR and its black-box methods counterparts, *e.g.*, PINNs, and is a necessary step for engineering use cases.

$$u(x) = 2.0833 \times 10^{-6} x^4 - 4.166 \times 10^{-5} x^3 + 2.0833 \times 10^{-3} x. \quad (7)$$

This Euler-Bernoulli problem was run with $n = (2, 3, 11)$ input training data pairs (x_i, y_i) to assess the relative importance of physics-regularization to provided training data quantity. In the case of $n = 2$ training points, this represents specification of only the boundary conditions stated for $u(x)$. This numerical experiment is extended to a single training point at the center of the beam for $n = 3$ training points and ultimately to a training point at each unit distance along the beam with $n = 11$

training points. For each of the physics-regularized trials, $x^{(1)} \in (0, l)$ and $x^{(2)} \in (1, 2, \dots, 9)$: a total of $m = 11$ physics-regularization points.

As a baseline performance comparison for physics-regularized GPSR, conventional GPSR was also completed. For the conventional GPSR trials, training data sizes of $n = (3, 5, 11)$ were tested. Because the case of $n = 2$ training points could not be carried out with conventional GPSR (as the solution would be trivial, $f(x) = 0$) a case of $n = 5$ training points was added to illustrate data quantity-dependent performance behavior.

Table 1: GPSR hyperparameters for solving the Euler-Bernoulli equation.

Hyperparameters		Value(s)
Operator	Test 1	$+, -, \times$
	Test 2	$+, -, \times, pow, sin, \div$
Number of islands		10
Population size		150
Maximum complexity		10
Crossover rate		0.5
Mutation rate		0.5
Differential weight, λ		1
Evolutionary algorithm		Deterministic crowding

Because of the stochastic evolutionary process inherent in GP, each of these cases was repeated 30 times to determine average behavior. For each of these numerical experiments the hyperparameters listed in Table 1 were used. These hyperparameters were not chosen to optimize the performance of GPSR for this case. Instead, generic defaults were used *e.g.*, 50% crossover vs. mutation, to demonstrate functionality in a generic sense, *i.e.*, without a bias toward calibrated hyperparameters. Further, we test dependence of successful solution evolution on the user-defined mathematical building blocks (operators). In Test 1, operators are limited to the minimal requisite set from which the solution can be evolved. In contrast, Test 2 permits an equal number of unnecessary operators, which expands the models search space for GPSR.

3.2. Poisson's Equation

The next test problem is an elliptic PDE boundary value problem known as Poisson's equation, which has a variety of applications in theoretical physics Evans (2010);

Han and Lin (2011). The objective of this experiment is to evolve the known closed-form solution using physics-regularized GPSR. Additionally, we investigate the influence of domain dimensionality, *i.e.*, one, two, or three dimensional (1D, 2D, 3D, respectively) and permitted operators (similar to the Euler-Bernoulli experiment) on the performance of physics-regularized GPSR. Poisson's equation with Dirichlet boundary conditions is defined as:

$$L^{(1)}(u(x)) = \nabla^2 u(x) = f(x) \quad \text{in } \Omega \quad (8a)$$

$$u(x) = 0 \quad \text{on } \partial\Omega, \quad (8b)$$

where $u(x)$ is the solution on $x \in (0, 1)^d$, $f(x)$ is a source term defined by $-d \pi^2 \prod_{i=1}^d \sin(\pi x_i)$ where $d \in (1, 2, 3)$, Ω is the problem domain, and $\partial\Omega$ is the boundary of the problem. Figure 2 is an example of the sampled coordinates in the 2D domain, where Equation 8a is evaluated at circles and Equation 8b is evaluated at triangles.

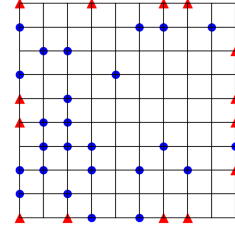


Figure 2: Example of the sampled data in the 2D domain

The solution to Equation 8 can be obtained through integration and application of boundary conditions, which leads to:

$$u(x) = \prod_{i=1}^d \sin(\pi x_i) \quad (9)$$

The analytical solutions for 1D, 2D and 3D then become Equation 10a, 10b and 10c, respectively. The resulting physics-regularized GPSR models were evaluated numerically and symbolically to confirm if the target solution was evolved:

$$u(x_1) = \sin(\pi x_1) \quad (10a)$$

$$u(x_1, x_2) = \sin(\pi x_1) \cdot \sin(\pi x_2) \quad (10b)$$

$$u(x_1, x_2, x_3) = \sin(\pi x_1) \cdot \sin(\pi x_2) \cdot \sin(\pi x_3). \quad (10c)$$

A numerical experiment was run with randomly sampled coordinates along the boundary, $n = 2$ for 1D, $n = 16$ for 2D, $n = 20$ for 3D, for evaluation of Equation 8b, and along the domain, $m = 2$ for 1D, $m = 32$ for 2D, $m = 64$ for 3D, for evaluation of Equation 8a, see Figure 2. This tests the minimal amount of input information for a well-posed problem.

Table 2: GPSR hyperparameters for solving the Poisson’s equation.

Hyperparameters		Value(s)
Operator	Test 1	\times , \sin
	Test 2	$+$, $-$, \times , \div , \sin , \cos
Number of islands		10
Population size		150
Maximum complexity		20
Crossover rate		0.5
Mutation rate		0.5
Differential weight, λ		1
Evolutionary algorithm		Deterministic crowding

For each of these numerical experiments the hyperparameters listed in Table 2 were used to compare performance. As with the Euler-Bernoulli experiment, the number of islands, population size, crossover rate, mutation rate, differential weight, and evolutionary algorithm were not tuned in an attempt to demonstrate a generic, default performance. Also, tested in this experiment is the dependence of performance on user-defined mathematical building blocks (operators). In Test 1, operators are limited to the minimal requisite set from which the solution can be evolved. Additionally, Test 2 permits a $3\times$ number of unnecessary operators.

4. Results

Each of the numerical experiments presented in this section were run 30 times to gather variation in results due to the inherent stochasticity of the genetic programming algorithm. Each GPSR trial was terminated upon evolving a model for which $F \leq 1 \times 10^{-15}$, see Equation 4. Of particular interest here is the ability of GPSR to evolve the known (correct) equation, not just a numerically-accurate approximation to that equation. Here, a correct equation is defined as being an algebraic equivalent to the known

solution and a trial was only considered to be successful if such an equation evolved. Consequently, success rates pertain to the frequency at which the correct analytical solution is determined, and not by simply reaching the prescribed fitness threshold.

4.1. Euler-Bernoulli equation

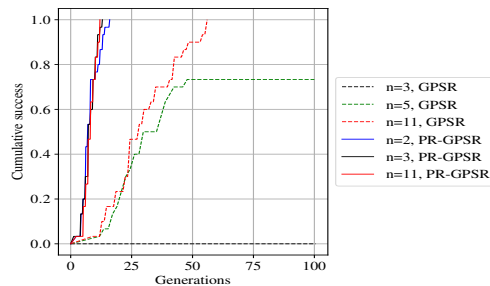


Figure 3: Cumulative distribution of successful (*i.e.*, produced known model) GPSR model evolution.

To compare the performance of physics-regularized GPSR to conventional GPSR the cumulative distribution of required generational counts to achieve the correct equation is illustrated in Figure 3. In this experiment, the number of points, n , used to evaluate the fitness contribution from Equation 1 is varied to quantify the effect of available training data on successful evolution of the known solution. For the physics-regularized trials, the number of points, m , used to evaluate the fitness contribution from Equation 3 is held fixed at 11, unless otherwise specified.

The inclusion of physics-regularization reliably evolved the known solution is shown in Figure 3, and the corresponding performance (number of generations to success) was significantly improved. With the conventional fitness function GPSR was able to reliably determine the known solution if $n = 11$ training points were provided, while with $n = 5$ the known solutions was found about 75% of the time and with $n = 3$ the known solution was never evolved. In the cases of conventional fitness with $n = 3$ or $n = 5$ training points, GPSR produced models with low fitness but with equations that only fit those few data points but elsewhere were poor.

To test and illustrate a general measure of accuracy, Figure 4 contains the average (of the 30 repeats) fitness

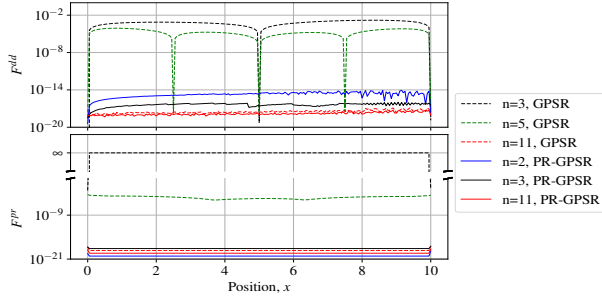


Figure 4: Test fitness evaluations at 201 points in x decomposed into (top) F^{dd} and (bottom) F^{Pr} .

terms along the beam, x , for each studied case. First, the error from provided displacement training data, F^{dd} , are plotted at 201 points (steps of 0.05) along the beam in Figure 4 (top). As can be seen for the $n = 3$ and $n = 5$ training points cases with conventional GPSR, the produced models are only accurate specifically at the training points. Elsewhere, those two cases produce models that are especially inaccurate. All other tested cases, however, produce numerically-accurate models: producing highly accurate models at points that were not specifically provided as training data.

Next, a test for general model accuracy for the physics-regularized term, F^{Pr} , is presented in Figure 4 (bottom). For models that resulted in undefined F^{Pr} evaluation *e.g.*, were not fourth-order differentiable, the fitness was set to ∞ . These results are consistent with results of Figure 3, where the combination of conventional fitness and low data quantity (*i.e.*, $n = 3$ or $n = 5$ data points) did not reliably produce the known solution. All other cases resulted in negligible test errors for both F^{dd} and F^{Pr} . For these cases, an algebraically-equivalent form of the known solution was evolved and the non-zero error is a direct consequence of the finite error threshold that results from the local optimization step, Section 2.3, and round-off error during model evaluation.

In both tests for general model accuracy, F^{dd} and F^{Pr} , the physics-regularized cases worked well and performance was relatively insensitive to the quantity of training data provided. In other words, even in the case that only the boundary conditions (*i.e.*, $n = 2$ training points) are provided, physics-regularized GPSR produces the known solution. By contrast, with the conventional fitness, GPSR

only reliably produced the known solutions with $n = 11$ training points. Further, from these results it is observed that the incorporation of physics-regularized fitness is even more beneficial than added training data. Finally, beyond these numerical evaluations, symbolic regression affords the unique opportunity to evaluate the model form, algebraically. These evaluations are provided in Section 5 along with the implications that these results suggest for engineering use of physics-regularized, interpretable ML methods.

As provided in Table 1, the preceding tests were run with $+$, $-$, and \times as the permitted operators for GPSR. And, while that represents the minimal set of operators for this experiment it is important to understand how unneeded operators might affect GPSR performance. Consequently, the same numerical experiment was repeated (again 30 times), but with more operators (annotated as MO): *pow*, *sin*, and \div .

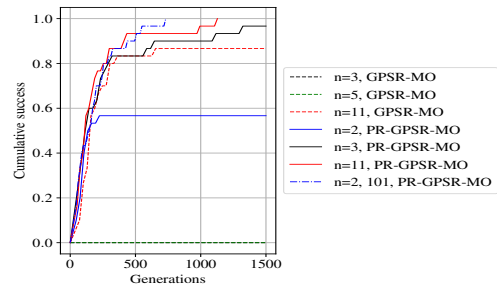


Figure 5: Cumulative distribution of successful GPSR model evolution with unneeded operators permitted.

For these tests with unneeded operators a corresponding shift in behavior is presented in Figure 5. First, it is seen that the cases of conventional fitness with $n = 3$ or $n = 5$ training points were never observed to produce the known solutions while with 11 training points the correct solution was evolved only approximately 85% of the time and required approximately 20 \times the number of generations as compared Figure 3. Next, unlike with the previous case with a minimum operator set, the physics-regularized cases now demonstrate a clear dependence of success on training data quantity. Here, the case of two training data points (BCs only) now produces the known solution just over half the time and approximately 95% with three points. Extending to $n = 11$ training points re-

liably produces the known solution, but requires approximately 100x the generations. An additional case was assessed here, with $n = 101$ training points, which demonstrates continued performance improvement in the requisite generation count for determination of the known solution, but still requires an order of magnitude more generations than illustrated in Figure 3. This qualitative shift of performance in reproducing the known solution was determined to mainly be a result of including the \sin operator. For evolved models that included \sin , F^{Pr} was defined (*i.e.*, always fourth-order differentiable) and the local optimization was consistently successful in finding model parameters that produced a low-fitness model. However, while these models were numerically accurate, they were not algebraically equivalent to the known solution and therefore did not constitute success in these tests. Clearly, the included operators play a significant role in GPSR performance even with the inclusion of a physics-regularized fitness function.

4.2. The Poisson's Equation

A plot of 30 GPSR trials is shown in Figure 6 to visualize the influence of domain dimensionality and permitted operators on the performance of the physics-regularized GPSR implementation. For consistency, each result is represented as the same color in all sub plots. The solid lines correspond to median values and the darker and lighter colored section correspond to the quartile range and outliers, respectively. From Figure 6, it is observed that all cases achieved a fitness below the specified threshold, $F \leq 1 \times 10^{-15}$. Further, in each of the trials, the known solution was successfully evolved. And generally, as would be expected, the requisite number of generations increased significantly with problem dimensionality.

The effect of domain dimensionality on the performance with only two operators (\sin and \times) is presented in Figure 6a. It can be seen that 1 generation was required for 1D (not illustrated and not surprising due to the low complexity of the solution), 2 generations for 2D, and 28 generations for 3D. The effect of the unneeded operators, $+$, $-$, \div and \cos , is illustrated in Figure 6b. Overall, the trend is similar to Figure 6a, but requires significantly more generations for success. This comparison is made more clear by Figures 6c and 6d. In each of these tests, lower dimensions tended to result in lower fitness. This

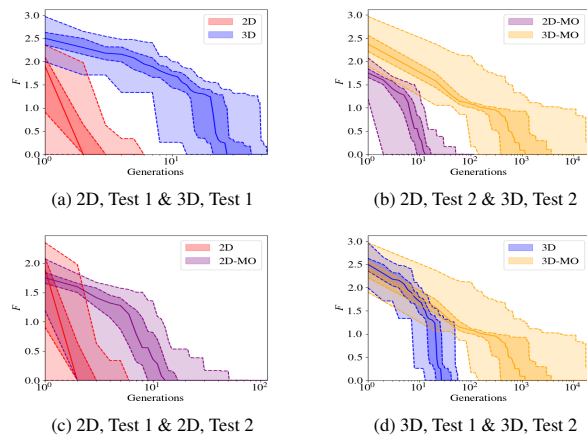


Figure 6: Fitness results of 30 SR runs for all trials. The solid line is the median fitness at each generation. The two filled regions of each color represents the quartile range (darker) and outliers (lighter), respectively.

result was found to be due to fewer parameters to be determined during local optimization and round-off error.

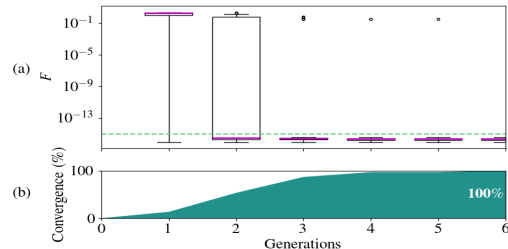


Figure 7: Successful evolution of known solution over generations in the 2D, Test 1 trial. (a) fitness evolution as box plot of most fit individuals in each population across the 30 runs. The median is illustrated by the purple line, and outliers by circles. The numerical convergence threshold is shown as the green dashed line. (b) cumulative distribution of success.

The statistical effect of dimensionality and included operators on success rates is illustrated in Figures 7-10. In each, the box plot illustrates the distribution of best fit individuals in the population across all 30 runs of each trial (combined 2D or 3D, and operator set). The boxes extend from the lower to upper quartile with whiskers extending to show the distribution range, and outliers shown as open circles. For the 2D trial with only necessary operators included (Test 1), Figure 7, it is seen that convergence

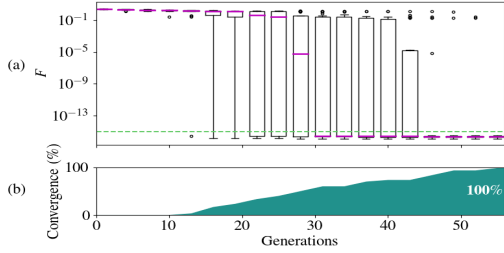


Figure 8: Successful evolution of known solution over generations in the 3D, Test 1 trial. (a) fitness evolution as box plot of most fit individuals in each population across the 30 runs. The median is illustrated by the purple line, and outliers by circles. The numerical convergence threshold is shown as the green dashed line. (b) cumulative distribution of success.

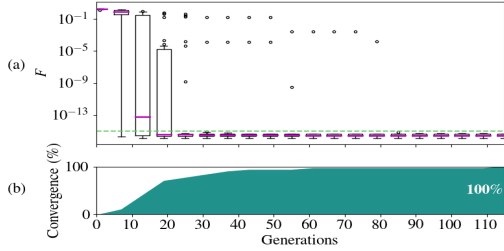


Figure 9: Successful evolution of known solution over generations in the 2D, Test 2 trial. (a) fitness evolution as box plot of most fit individuals in each population across the 30 runs. The median is illustrated by the purple line, and outliers by circles. The numerical convergence threshold is shown as the green dashed line. (b) cumulative distribution of success.

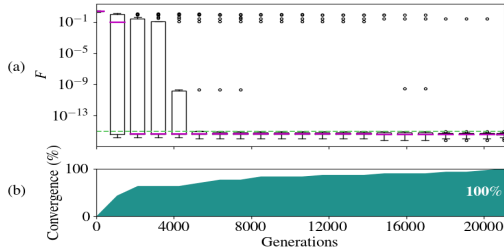


Figure 10: Successful evolution of known solution over generations in the 3D, Test 2 trial. (a) fitness evolution as box plot of most fit individuals in each population across the 30 runs. The median is illustrated by the purple line, and outliers by circles. The numerical convergence threshold is shown as the green dashed line. (b) cumulative distribution of success.

occurs quickly with all 30 runs producing the known solution by the sixth generation. With the same set of permitted operators (Test 1), but extended to 3D, a significant ($\sim 10\times$) increase in the number of required generations to reliably evolve the known solution is illustrated in Figure 8. Next, the effect of an increased set of permitted operators (Test 2) is presented in Figures 9 and 10. For the 2D case with increased operators, approximately 100 generations were required for successful evolution of all 30 runs as presented in Figure 9. Lastly, the most complex case of 3D with increased operators, Figure 10, required approximately 20k generations. It should be noted that these generation counts represent the requirement to achieve success across all 30 runs. However, in each tested case, 1-2 outliers caused a significant increase in the number of generations required for 100% success.

It was anticipated that higher dimensionality would increase requisite generations for success, as illustrated. However, data illustrated in Figures 7-10 highlights that the overall performance was more sensitive to the inclusion of additional, unneeded operators in GPSR than the increase in problem dimensionality. To assess the reason underpinning this observation consider that each GPSR model consists of d , m , and n , where d is the dimension, m is the number of operators, and n is the maximum complexity, respectively. An estimate of the total number of potential models in a search space is given by \mathcal{H} :

$$\mathcal{H} = (2d + m)^{n-2d} \prod_{i=1}^{2d} (n - i + 1) (2d - i + 1) \quad (11)$$

$$\approx O((2d + m)^{n-2d} (2nd)^{2d}).$$

Consequently, the size of hypothesis space can be estimated for the 2D cases as $O(1.2 \times 10^{20})$ with minimal operators and $O(4.1 \times 10^{23})$ for additional operators. Similarly, for the 3D cases, the space is estimated as $O(1.3 \times 10^{25})$ for minimal operators, and $O(3.8 \times 10^{27})$ for added operators. Overall, this estimates the relative influence of the increased dimensionality and operator set on the GPSR search space and, as a consequence, ability to find the correct solution.

5. Discussion

As was the goal of including the physics-regularized fitness function in GPSR, the preceding results demonstrate

meeting the more stringent ability for mathematical verification *i.e.*, determination of known analytical models. This demonstration is especially important in the engineering and science contexts due to the resulting explainability of produced models and the trust that can be built. In this final section, we discuss various aspects of the results specifically related to observation of evolved equation forms for the GPSR experiments.

5.1. Euler-Bernoulli equation

As can be inferred from Equation 7 the simplest, correct GPSR model would be of the form:

$$\alpha \cdot x^4 + \beta \cdot x^3 + \gamma \cdot x, \quad (12)$$

where the following represent the global minimum to be determined during local calibration: $\alpha = 2.08\bar{3} \times 10^{-6}$, $\beta = -4.1\bar{6} \times 10^{-5}$, and $\gamma = 2.08\bar{3} \times 10^{-3}$.

However, there are an infinite number of possibilities in which equivalent model forms evolve within GPSR. Assessment of equivalency with the correct model then requires additional efforts in automating GPSR model simplification. For this, the python sympy module was utilized Meurer et al. (2017). In all of the Euler-Bernoulli trials we observed that an algebraically-equivalent model form, to Equation 12, evolved in cases where the general fitness test, see Figure 4, was $< 1 \times 10^{-15}$ across the 201 test points. Note that this threshold was found to be conservative but ensured no false positives in the verification tests. Additionally, in the case of the minimal set of operators, physics-regularized GPSR produced the correct model form and coefficients reliably while conventional GPSR produced the correct model only if sufficient training data were provided for determination of the necessary 4th-order polynomial. One such observed example is:

$$(b \cdot d \cdot f) \cdot x^4 + (b \cdot d \cdot e) \cdot x^3 + (b \cdot c) \cdot x + a. \quad (13)$$

In which case, local optimization can successfully return:

$$\begin{aligned} b \cdot d \cdot f &= \alpha = 2.08\bar{3} \times 10^{-6}, \\ b \cdot d \cdot e &= \beta = -4.1\bar{6} \times 10^{-5}, \\ b \cdot c &= \gamma = 2.08\bar{3} \times 10^{-3}, \text{ and} \\ a &= 0. \end{aligned} \quad (14)$$

Upon analyzing the evolved models that had not yet achieved the specified fitness threshold, it was commonly observed that the correct model form had evolved but with more complex interrelationships among the coefficients. In these cases, successful local optimization as given in Equations 14 was precluded. For example, the GPSR-produced model:

$$(b^2 \cdot d) \cdot x^4 + (2 \cdot a \cdot b \cdot d) \cdot x^3 + (a^2 \cdot d) \cdot x^2 + (a \cdot c) \cdot x \quad (15)$$

would have the correct model form if the x^2 coefficient, $a^2 \cdot d$, evaluates to zero. However, for this to be the case either a or d must be zero, which would mean that the coefficient of x^3 must also become zero, ultimately leading to an incorrect model. For improved use in engineering and science applications, an evolutionary mechanism that algebraically simplifies the model and aggregates these redundant coefficient relationships could improve performance and improve interpretability, by reducing unnecessary complexity of the evolved models. However, such simplifications would also alter evolutionary paths.

For the physics-regularized GPSR tests with two training points and unneeded operators included, incorrect model forms like the following were often observed:

$$a + b \cdot \sin(c + \sin(x/c - d + e/c)). \quad (16)$$

In these specific cases, sinusoidal models commonly resulted in low fitness due to the even spacing of the training points, X_i , and differential equation evaluation points, X_j . It was observed that increases in either X_i or X_j , along with selecting points randomly within the problem domain precluded these misleading models during evolution. Consequently, a convergence test is useful in determining a true model form, *i.e.*, assessment of the sensitivity of model form to added X_i or X_j points. Upon convergence in this context, model forms among repeated GPSR trials should reliably contain consistent operators.

5.2. Poisson's Equation

The efficacy of physics-regularized GPSR for evolving analytic solutions of PDEs is investigated using the Poisson's equation. In this experiment, focus is on the influence of domain dimensionality and permitted operators on successful model evolution. First, it is found that

the physics-regularized GPSR implementation successfully obtained the correct symbolic equation for all problem dimensions tested, such as Equation 10a, 10b, and 10c. However, as is also observed in the Euler-Bernoulli experiment, a variety of equivalent model forms evolved, which are commonly of higher complexity than the simple form:

$$\tilde{u}(x) = \prod_{i=1}^d a_i \cdot \sin(b_i x_i), \quad (17)$$

$$a_i = 1, b_i = \pi.$$

Further, as the dimensionality increased, higher complexity versions of the true model became more commonly observed. Of these cases, forms that can be equivalent to the known solution, upon successful local optimization, were commonly observed:

$$\tilde{u}(x) = \prod_{i=1}^d a_i \cdot \sin(b_i x_i + c_i), \text{ where} \quad (18)$$

$$a_i = 1, b_i = \pi, c_i \approx 0 \text{ or}$$

$$\tilde{u}(x) = \prod_{i=1}^d a_i \cdot \cos(b_i x_i + c_i), \text{ where} \quad (19)$$

$$a_i = 1, b_i = \pi, c_i = -\pi/2.$$

Unlike the Euler-Bernoulli experiment, however, this experiment always resulted in model forms *e.g.*, Equations 18 and 19, for which the parameters could be correctly determined upon local optimization. Generally, upon assessment of the various model forms in both experiments, it is evident that model bloat was increased in the polynomial solutions of the Euler-Bernoulli experiment.

6. Conclusions

The developed combination of an interpretable ML method, GPSR, with fitness regularized by known governing differential equations, is demonstrated to discover their analytical solutions. This method is demonstrated on two governing differential equations, a fourth-order ODE

and second-order PDE, where the known analytical solution is successfully discovered in both cases. Because of the inherent interpretability, determination of success is defined as learning the true algebraic solution, and not limited to a numerical approximation, as is more conventional ML methods.

The physics-regularized GPSR method requires only a statement of the (known) governing differential equation and boundary conditions sufficient for a well-posed problem, without need for additional training data to successfully determine the solutions. However, the success of the physics-regularized GPSR method is demonstrated to be dependent on several factors. In the Euler-Bernoulli problem (fourth-order ODE), the analytical solution was determined in each of the 30 repeated trials when only the requisite mathematical operators were permitted. However, upon expanding to twice the number of operators to include unnecessary operators, it is found that a minimal amount of training data points were required to guide the physics-regularized GPSR to the correct model form. In other words, in the practical case where the operators in the solution to the differential equation are unknown it is still likely that some amount of (albeit a significantly reduced quantity) training data will be needed to identify the true model form. In the Poisson problem (second-order PDE), the effect of dimensionality was tested. As expected for this case, the required number of generations to discover the true solution was significantly increased with dimension. Nevertheless, the analytical solution was recovered in every trial.

The ability to use interpretable ML to discover analytical solutions to known differential equations has broad application within computational mechanics. Determination of an analytical expression, as opposed to numerically-approximated data, enables broader mathematical treatments *e.g.*, sensitivity studies, optimization, and deeper insights into the solution characteristics, to name a few. While the method presented here provides the first necessary step in applied mechanics and engineering, verification, the next step is to employ this method for cases with known differential equations but unknown solutions. A key to building trust within the engineering community for using ML in practice will then be the demonstration of how one can validate those solutions.

Acknowledgements

The support and resources from the Center for High Performance Computing at the University of Utah are gratefully acknowledged. This research was sponsored in part by the Army Research Laboratory (ARL) under Cooperative Agreement Number W911NF-12-2-0023 and by Sandia National Laboratories under Agreement 2262518. The second and fourth authors were partially supported by the Defense Advanced Research Projects Agency (DARPA) through the Transformative Design (TRADES) program under the award HR0011-17-2-0016. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARL or the US Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Bibliography

References

- G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nature Reviews Physics* 3 (2021) 422–440. doi:<https://doi.org/10.1038/s42254-021-00314-5>.
- M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics* 378 (2019) 686–707. doi:<https://doi.org/10.1016/j.jcp.2018.10.045>.
- A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, V. Kumar, Theory-guided data science: A new paradigm for scientific discovery from data, *IEEE Transactions on knowledge and data engineering* 29 (2017) 2318–2331. doi:<https://doi.org/10.1109/TKDE.2017.2720168>.
- M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561 (2017). doi:<https://doi.org/10.48550/arXiv.1711.10561>.
- N. M. Heckman, T. A. Ivanoff, A. M. Roach, B. H. Jared, D. J. Tung, H. J. Brown-Shaklee, T. Huber, D. J. Saiz, J. R. Koepke, J. M. Rodelas, J. D. Madison, B. C. Salzbrenner, L. P. Swiler, R. E. Jones, B. L. Boyce, Automated high-throughput tensile testing reveals stochastic process parameter sensitivity, *Materials Science and Engineering: A* 772 (2020) 138632. doi:<https://doi.org/10.1016/j.msea.2019.138632>.
- J. E. Warner, J. Cuevas, G. F. Bomarito, P. E. Leser, W. P. Leser, Inverse estimation of elastic modulus using physics-informed generative adversarial networks, arXiv preprint arXiv:2006.05791 (2020). doi:<https://doi.org/10.48550/arXiv.2006.05791>.
- A. Adadi, M. Berrada, Peeking inside the black-box: a survey on explainable artificial intelligence (xai), *IEEE access* 6 (2018) 52138–52160. doi:<https://doi.org/10.1109/ACCESS.2018.2870052>.
- F. K. Došilović, M. Brčić, N. Hlupić, Explainable artificial intelligence: A survey, in: 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO), IEEE, 2018, pp. 0210–0215. doi:<https://doi.org/10.23919/MIPRO.2018.8400040>.
- C. Rudin, J. Radin, Why are we using black box models in ai when we don't need to? a lesson from an explainable ai competition, *Harvard Data Science Review* 1 (2019). doi:<https://doi.org/10.1162/99608f92.5a8a3a3d>.
- D. Versino, A. Tonda, C. A. Bronkhorst, Data driven modeling of plastic deformation, *Computer Methods in Applied Mechanics and Engineering* 318 (2017) 981–1004. doi:<https://doi.org/10.1016/j.cma.2017.02.016>.
- G. Bomarito, T. Townsend, K. Stewart, K. Esham, J. Emery, J. Hochhalter, Development of interpretable, data-driven plasticity models with symbolic regression, *Computers & Structures* 252 (2021) 106557. doi:<https://doi.org/10.1016/j.compstruc.2021.106557>.

- A. Hernandez, A. Balasubramanian, F. Yuan, S. A. Mason, T. Mueller, Fast, accurate, and transferable many-body interatomic potentials by symbolic regression, *npj Computational Materials* 5 (2019) 1–11. doi:<https://doi.org/10.1038/s41524-019-0249-1>.
- M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, *science* 324 (2009) 81–85. doi:<https://doi.org/10.1126/science.1165893>.
- J. R. Koza, J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, volume 1, MIT press, 1992.
- M. Schmidt, H. Lipson, Symbolic regression of implicit equations, in: *Genetic Programming Theory and Practice VII*, Springer, 2010, pp. 73–85. doi:https://doi.org/10.1007/978-1-4419-1626-6_5.
- Y. Wang, N. Wagner, J. M. Rondinelli, Symbolic regression in materials science, *MRS Communications* 9 (2019) 793–805. doi:<https://doi.org/10.1557/mrc.2019.85>.
- NASA, Bingo, <https://github.com/nasa/bingo>, 2013.
- D. L. Randall, T. S. Townsend, J. D. Hochhalter, G. F. Bomarito, Bingo: A customizable framework for symbolic regression with genetic programming, in: *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '22 Companion*, ACM, Boston, MA, USA, 2022. doi:10.1145/3520304.3534031.
- S. W. Mahfoud, Niching methods for genetic algorithms, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1995.
- F. Fernandez, G. SPEZZANO, M. Tomassini, L. Vaneschi, 6 parallel genetic programming, *Parallel metaheuristics: a new class of algorithms* (2005) 127.
- D. Whitley, S. Rana, R. B. Heckendorn, The island model genetic algorithm: On separability, population size and convergence, *Journal of computing and information technology* 7 (1999) 33–47.
- A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, *arXiv preprint arXiv:1912.01703* (2019). doi:<https://doi.org/10.48550/arXiv.1912.01703>.
- N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, J. Reynolds, A. Melnikov, N. Lunova, O. Reblitz-Richardson, Pytorch captum, <https://github.com/pytorch/captum>, 2019.
- P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Í. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy: Open source scientific tools for Python, <http://www.scipy.org/>, 2001.
- V. V. De Melo, B. Fowler, W. Banzhaf, Evaluating methods for constant optimization of symbolic regression benchmark problems, in: *2015 Brazilian conference on intelligent systems (BRACIS)*, IEEE, 2015, pp. 25–30. doi:<https://doi.org/10.1109/BRACIS.2015.55>.
- L. C. Evans, *Partial differential equations*, volume 19, American Mathematical Soc., 2010. doi:<http://dx.doi.org/10.1090/gsm/019>.
- Q. Han, F. Lin, *Elliptic partial differential equations*, volume 1, American Mathematical Soc., 2011. doi:<https://doi.org/10.1090/cln/001>.
- A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Korpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, A. Scopatz, Sympy: symbolic computing in python, <https://docs.sympy.org/>, 2017.