fluids

MDPI

Article

Equation Discovery Using Fast Function Extraction: a Deterministic Symbolic Regression Approach

Harsha Vaddireddy and Omer San *

School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK 74078, USA; hvaddir@okstate.edu

* Correspondence: osan@okstate.edu; Tel.: +1-405-744-2457; Fax: +1-405-744-7873

Received: 27 April 2019; Accepted: 12 June 2019; Published: 15 June 2019

Abstract: Advances in machine learning (ML) coupled with increased computational power have enabled identification of patterns in data extracted from complex systems. ML algorithms are actively being sought in recovering physical models or mathematical equations from data. This is a highly valuable technique where models cannot be built using physical reasoning alone. In this paper, we investigate the application of fast function extraction (FFX), a fast, scalable, deterministic symbolic regression algorithm to recover partial differential equations (PDEs). FFX identifies active bases among a huge set of candidate basis functions and their corresponding coefficients from recorded snapshot data. This approach uses a sparsity-promoting technique from compressive sensing and sparse optimization called pathwise regularized learning to perform feature selection and parameter estimation. Furthermore, it recovers several models of varying complexity (number of basis terms). FFX finally filters out many identified models using non-dominated sorting and forms a Pareto front consisting of optimal models with respect to minimizing complexity and test accuracy. Numerical experiments are carried out to recover several ubiquitous PDEs such as wave and heat equations among linear PDEs and Burgers, Korteweg–de Vries (KdV), and Kawahara equations among higher-order nonlinear PDEs. Additional simulations are conducted on the same PDEs under noisy conditions to test the robustness of the proposed approach.

Keywords: deterministic symbolic regression; fast function extraction; compressive sensing; pathwise regularized learning; non-dominated sorting

1. Introduction

Partial differential equations (PDEs) are ubiquitous in all branches of science and engineering. These mathematical models are generally derived from conservation laws, sound physical arguments, and empirical heuristics drawn from experiments by an insightful researcher. However, there are many complex systems (some being neuroscience, weather forecasting, disease control modeling, finance, and ecology) that are still awaiting quantitative models to be built on physical arguments. The counter approach is to build models based on observing complex patterns in enormous amounts of readily available data which can loosely be termed as reverse-engineering nature. This approach has been used in previous studies, for example by Kepler, to approximate elliptic orbits solely from planetary positions. This reverse-engineering is appropriate today as we can leverage computers to identify patterns from observing data that might not be comprehensible to humans. Recent progress in machine learning and data science [1,2], combined with an increase in computational power, has generated innovative algorithms to distill physical models from data. These algorithms are termed as data-driven tools, as they can infer correlations and extract patterns from high-dimensional data or measurements. Some popular data-driven tools for extracting dynamical system from data are artificial

neural networks, compressive sensing, sparse optimization, and symbolic regression approaches based on evolutionary algorithms.

Artificial neural network (ANN) is a branch of machine learning also referred to as deep learning (if multiple hidden layers are used), a technique that transforms input features through nonlinear interactions and maps to output target features [3,4]. ANNs emulate human brain functioning with several hidden layers consisting of so-called neurons with specific weights assigned to each of them. These ANN models have gained popularity in recent times due to their superior performance in modeling complex nonlinear interactions across a wide range of applications including image processing [5], video classification [6] and autonomous driving [7]. Extreme learning machine (ELM) was used as auto-encoders to extract the basis for the reconstruction of fine-scale information from limited data [8]. The major drawback with the approach of a deep-learning framework is that the obtained model is not quite open to physical inference or interpretability and in general is considered to be a black-box model.

Compressive sensing (CS) [9,10] has been applied to signal processing in seeking the sparsest solution (i.e., solution with the fewest number of non-zero basis functions from data). Sparsity-promoting optimization techniques [11,12] play fundamental roles in CS and are generally categorized under basis pursuit algorithms [13]. In sparse optimization, the least squares (LS) objective function is regularized in such a way that additional constraints on the cost function are added to shrink large coefficients and avoid overfitting, thereby promoting sparsity in feature selection. The least absolute shrinkage and selection operator (LASSO) [11,14] is one of the most popular regularized LS regression methods. LASSO performs feature selection through L_1 penalty added to LS objective function to recover sparse solutions [15]. Ridge regression [16] is another regularized variant where L_2 penalty is added to LS objective function. The L_2 penalty helps in grouping multiple correlated basis functions and increases robustness and convergence stability for ill-conditioned systems. Elastic net approach [17,18] is a hybrid of LASSO and ridge approach, combining the strengths of both algorithms.

Derived from these advances, sequentially threshold least squares (STLS) algorithm [19] was developed where a hard threshold on non-zero coefficients was performed recursively to find sparse solutions. This algorithm was leveraged to form a framework called sparse identification of nonlinear dynamics (SINDy) [19] to extract ordinary differential equations representing the underlying phenomena. This seminal work re-envisioned the model discovery from the preservative of sparse optimization and compressive sensing. They have recovered various benchmark dynamical systems such as chaotic Lorenz system and vortex shedding behind the cylinder. However, this framework faces challenge in recovering spatio-temporal data or high-dimensional measurements and highly correlated basis functions. This limitation was addressed using sequential threshold ridge regression (STRidge) algorithm forming a framework called PDE functional identification of nonlinear dynamics (PDE-FIND) [20]. PDE-FIND was applied to high-dimensional spatio-temporal measurements representing various nonlinear dynamics. This framework also performs reasonably well under addition of noise to data/measurements. The sparse optimization methods discussed above generally have a free parameter associated with penalty term that is tuned by the user to recover multiple models containing complex parsimonious models.

Other works exploiting L_1 regularized LS minimization were used to recover various nonlinear PDEs [21,22] using both high-fidelity and distorted (noise) data. Additionally, limited and distorted data samples were used to recover chaotic and high-dimensional systems [23,24]. Bayes information criteria was used to rank different recovered models with different complexity for completely new systems, thereby realizing confidence in the recovered model [25]. The above discussed frameworks assume that the structure of the model to be recovered is sparse in nature; that is, only a small number of terms govern the dynamics of the system. This assumption holds for many physical systems in science and engineering.

Symbolic regression (SR) approaches based on evolutionary computation [26,27] are another class of frameworks that can find analytically tractable functions that have been applied to system

identification problems. Traditional linear and nonlinear regression algorithms assume a mathematical form and only find parameters that best fit the data. On the other hand, SR approaches aim to simultaneously find parameters and learn functional form of the model from observed data. This is generally achieved by searching mathematical abstractions with a preselected set of arithmetic operators while minimizing the error metrics. Finally, optimal model is selected from Pareto front analysis with respect to minimizing accuracy versus model complexity. Genetic programming (GP) [26] is a popular choice leveraged by most of the SR frameworks. GP is an evolutionary computation technique that is inspired by Darwin's theory of natural evolution. A seminal work was done in extracting nonlinear dynamics [28] from input–output response using the GP approach. The use of GP-based SR approaches has appeared in various system identification problems [29–31]. Furthermore, GP has been applied to identify closed-loop feedback control for turbulent separated flows [32,33]. Different improved versions of GP have been proposed recently; for instance, gene expression programming (GEP) [27], parse matrix evolution (PME) [34], and linear genetic programming (LGP) [35]. GEP has been exploited recently to recover functional models approximating nonlinear behavior of stress tensors in Reynolds-averaged Navier–Stokes (RANS) and large eddy-simulation (LES) turbulence models [36,37]. Generally, GP-based SR approaches can identify models with complex nonlinear compositions given enough computational time.

Recently, researchers proposed fast and deterministic SR approaches by using results from CS and sparse optimization. Thus, a family of deterministic SR approaches were proposed. The first non-evolutionary SR algorithm was fast function extraction (FFX) [38] where feasible models were confined to generalized linear models (GLM) [39] and best bases and their corresponding coefficients were found by pathwise regularized learning that is also called elastic net algorithm [17]. The final models are selected through non-dominated filtering with respect to accuracy and model complexity. FFX draws influences from both GP and CS to distill better models from data. FFX solves quadratic optimization problems and thus, computation cost increases quadratically as we increase the number of bases in search space. FFX and GP has been applied to various problems such as dynamical system recovery and solar power prediction based on energy production data [40]. Elite base regression (EBR) [41] is a recent advancement in non-evolutionary computation where only elite bases are selected by measuring the correlation coefficient of basis functions with respect to the target model. These elite bases are spanned in search space and use the parse matrix encoding scheme to propagate the algorithm further to recover mathematical model. Prioritized grammar enumeration (PGE) [42] is another deterministic approach where genetic operators and random numbers from GP are replaced with grammar production rules and systematic choices. PGE approach also aims for the substantial reduction of search space.

In this paper, we demonstrate the use of FFX, a deterministic symbolic regression algorithm, to identify and recover the target PDEs representing both linear and nonlinear dynamical systems. We build candidate basis functions consisting of partial derivative terms of varying orders approximated by the central finite-difference formulas. For testing, we use exact analytical solutions of PDEs to get input data and use FFX Python package [38] to demonstrate its feasibility. First, we recover simple linear PDEs such as wave and heat equations. We then recover higher-order nonlinear PDEs such as Burgers, KdV, and Kawahara equations. We further add noise to input data originated from the same PDEs to test the robustness of FFX.

The rest of the paper is organized as follows. Section 2 gives a brief description of the FFX algorithm. In Section 3, FFX is tested on different canonical PDEs. We demonstrate performance and robustness of FFX by inferring dynamics from both clean and noisy input data. Section 4 gives the summary of our findings and limitations that need further investigation.

2. Methodology

Fast function extraction is a deterministic symbolic regression algorithm that draws its influences from compressive sensing and genetic programming. This method was predominately proposed to

improve speed and scalability while dealing with symbolic regression problems in deterministic sense. The following sections discuss the methodology used in the current paper for recovering PDEs using only data.

2.1. FFX Problem Statement

FFX aims to find white-box models or expressions, **Model** = $\{\text{model}_1, \text{model}_2, \ldots\}$ from a given set of data $\{\widetilde{\mathbf{\Theta}}, \mathbf{V}\}$. The input data $\widetilde{\mathbf{\Theta}}$ is arranged as a matrix with several rows equal to the number of input samples/measurements and several columns equal to the number of candidate feature/basis functions. The corresponding output data $\mathbf{V}$ is arranged as a column vector with several rows equal to the number of input samples/measurements. The recovered models undergo additional filtering to form a Pareto front consisting of optimum set with respect to tradeoff between minimizing model complexity f_1(model) and expected test error $f_2 = E_{x,y}L$(model), where L(model) is squared loss error of the model. In the current paper, we restrict the PDEs to be recovered to quadratic nonlinearity and up to fifth order in space. The general PDE to be recovered is in the form:

$$u_t = \mathscr{F}(\sigma, u, u^2, u_x, u_x^2, uu_x, u_{2x}, \ldots, u_{5x}^2) \tag{1}$$

where subscripts denote the order of partial differentiation and σ is an arbitrary parameter. For illustrative example, consider the problem of discovering PDE is the viscous Burgers equation as shown below:

$$u_t + uu_x = \nu u_{2x} \tag{2}$$

where $u(x,t)$ is generally velocity field and ν is viscous coefficient. The solution field $u(x,t) \in \mathbb{R}^{m \times n}$, where m is the number of time snapshots and n is the number of spatial locations, is formed by solving Equation (2) analytically or numerically. The solution field is arranged as shown below:

$$\mathbf{u} = \overbrace{\begin{bmatrix} u_1(t_1) & u_2(t_1) & \ldots & u_n(t_1) \\ u_1(t_2) & u_2(t_2) & \ldots & u_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_m) & u_2(t_m) & \ldots & u_n(t_m) \end{bmatrix}}^{\text{spatial locations}} \Bigg\} \text{time snapshots} \tag{3}$$

Higher-order partial derivative terms can be approximated using any available numerical schemes once recording of discrete data set given by Equation (3) is available. We use the leapfrog scheme for approximating the derivative in time, and the central difference schemes for space as follows:

$$\left.\begin{aligned} (u_t)_j^p &= \frac{u_j^{p+1} - u_j^{p-1}}{2dt} \\ (u_x)_j^p &= \frac{u_{j+1}^p - u_{j-1}^p}{2dx} \\ (u_{2x})_j^p &= \frac{u_{j+1}^p - 2u_j^p + u_{j-1}^p}{dx^2} \\ (u_{3x})_j^p &= \frac{u_{j+2}^p - 2u_{j+1}^p + 2u_{j-1}^p - u_{j-2}^p}{2dx^3} \\ (u_{4x})_j^p &= \frac{u_{j+2}^p - 4u_{j+1}^p + 6u_j^p - 4u_{j-1}^p + u_{j-2}^p}{dx^4} \\ (u_{5x})_j^p &= \frac{u_{j+3}^p - 4u_{j+2}^p + 5u_{j+1}^p - 5u_{j-1}^p + 4u_{j-2}^p - u_{j-3}^p}{2dx^5} \end{aligned}\right\} \tag{4}$$

where temporal and spatial step sizes are given by dt and dx, respectively. In Equation (4), the spatial location is denoted using subscript index j, and the temporal location using superscript index p.

We note that the other finite-difference schemes (or automated differentiation scheme) can be easily used within our study. FFX takes the input library consisting of basis functions (candidate terms) that are built using Equations (2) and (3). This core library is shown below:

$$\left.\begin{aligned} \mathbf{V(t)} &= \begin{bmatrix} \mathbf{U}_t \end{bmatrix} \\ \widetilde{\Theta}\mathbf{(U)} &= \begin{bmatrix} \mathbf{U} & \mathbf{U}_x & \mathbf{U}_{2x} & \mathbf{U}_{3x} & \mathbf{U}_{4x} & \mathbf{U}_{5x} \end{bmatrix} \end{aligned}\right\} \tag{5}$$

For each column in Equation (5), the solution $u(x,t)$ and its derivatives in space and time are arranged with size $m \cdot n \times 1$ where m is the number of time snapshots and n is the number of spatial locations. For example, the basis functions $\mathbf{U}$ and $\mathbf{U}_{2x}$ are arranged as follows:

$$\mathbf{U} = \begin{bmatrix} u(x_0,t_0) \\ u(x_0,t_1) \\ | \\ u(x_j,t_p) \\ | \\ u(x_n,t_m) \end{bmatrix}, \quad \mathbf{U}_{2x} = \begin{bmatrix} u_{2x}(x_0,t_0) \\ u_{2x}(x_0,t_1) \\ | \\ u_{2x}(x_j,t_p) \\ | \\ u_{2x}(x_n,t_m) \end{bmatrix} \tag{6}$$

where subscripts j and p denote the spatial and temporal collocation points of data, respectively. The basis functions in core library $\widetilde{\Theta}\mathbf{(U)}$ is expanded to include interacting basis functions (with itself and each other) limiting to quadratic nonlinearity. In addition, a constant term and original basis functions in $\widetilde{\Theta}\mathbf{(U)}$ are added to form the library of candidate features $\Theta\mathbf{(U)}$. The expansion procedure by FFX is discussed in Section 2.2. The final expanded library is in the form:

$$\Theta\mathbf{(U)} = \begin{bmatrix} \mathbf{1} & \mathbf{U} & \mathbf{U}^2 & \mathbf{U}_x & \mathbf{U}\mathbf{U}_x & \mathbf{U}_x^2 & \dots & \mathbf{U}_{5x}^2 \end{bmatrix} \tag{7}$$

where the size of library is $\Theta\ \mathbf{(U)} \in \mathbb{R}^{m \cdot n \times N_B}$, where N_B is number of basis functions (i.e., $N_B = 28$). For example, if we have 501 spatial points and 101 time snapshots with 28 candidate bases, then $\Theta\ \mathbf{(U)}$ (Equation (7)) contains 501×101 rows and 28 columns.

The Burgers PDE given by Equation (2) or any other PDE under consideration can be written in the form of linear system representation in terms of $\mathbf{V(t)}$ and $\Theta\mathbf{(U)}$:

$$\mathbf{V}(t) = \Theta\mathbf{(U)} \cdot \boldsymbol{\beta} \tag{8}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_{N_B}]$ is coefficient vector of size $\mathbb{R}^{N_B}$ where N_B is number of basis functions (candidate terms) in library $\Theta\mathbf{(U)}$. Thus, we form a linear system problem Equation (8) with over determined system (number of measurements greater than candidate terms). The goal of FFX is to find sparse coefficient vector $\boldsymbol{\beta}$ that only consists of active basis and rest of bases are set to zero. For example, in Burgers equation given by Equation (2), FFX has to find coefficient vector $\boldsymbol{\beta}$ consisting of only two basis functions, namely uu_x and u_{2x} and all others basis functions set to zero.

The system in Equation (8) can be solved for β using the LS problem. But LS minimization tries to fit all the basis functions resulting in all non-zero values in coefficient vector $\boldsymbol{\beta}$. Thus, solving Equation (8) using LS gives radically different functional form from the inferred dynamics and generally gives overfitted models. Hence sparse optimization is used which can identify the required basis functions along with coefficient estimation. FFX uses one of the sparse optimization algorithms called pathwise regularized learning (also known as elastic net approach) which is discussed in Section 2.3 for feature selection and parameter estimation. The sparse optimization in FFX returns multiple models of different complexity (number of basis functions). These models are further filtered using

non-dominated sorting inspired by genetic programming to extract Pareto optimal set of models with respect to minimizing complexity and test error. This filtering approach is discussed in Section 2.4. In summary, FFX algorithm performs three major steps to recover model from the data. They are:

- Construct a large library of linear and nonlinear basis functions. This step is known as basis function expansion or feature construction in ML.
- Use regularized least square algorithm to regress the coefficients and simultaneously select basis functions mapping to target model. FFX uses pathwise regularized learning also known as elastic net algorithm to get the vector of coefficients and corresponding bases that best fit the data. This step is termed as model building.
- Avoid over fitting using non-dominated filter (Pareto front analysis) with respect to model complexity (number of bases) versus the testing error. Final step is called model selection.

Please note that each of the above steps and associated algorithms elaborated in subsequent Sections 2.2–2.4 are derived from the original FFX algorithm [38] and only listed here for the sake of completeness. Readers are encouraged to refer the original algorithm [38] for in depth discussion.

2.2. Basis Function Expansion (Feature Construction)

FFX restricts the model form to be recovered to GLM [39]. GLMs are generalization on classic linear regression model. GLM aims to find $f(\vec{x})$ in a finite dimensional space spanned by a set of given basis functions as shown below:

$$f(\vec{x}) = \beta_0 + \sum_{i=1}^{N_B} \beta_i \theta_i(\vec{x}) \tag{9}$$

where θ_is are basis functions and β_is are corresponding coefficients. The important observation in Equation (9) is that the basis functions $\theta_i(\vec{x})$ are allowed to be nonlinear, but the whole ensemble process of $f(\vec{x})$ in GLM is considered linear combinations of these nonlinear basis functions.

The input library $\widetilde{\Theta}$ is passed to FFX where the library is expanded to Θ which consists of additional constant and nonlinear basis functions. FFX builds a massive set of nonlinear basis functions by interacting the bases with itself and each other limiting to quadratic nonlinearity. Additionally, it adds constant term and original basis functions to the expanded library. Algorithm 1 gives the outline of building interacting bases limiting to quadratic nonlinearity. In FFX, we can also construct uni-variate bases such as sqrt, exp, log, etc. But in our problem, we exclude this part as we restrict to only interacting bases. The OK function in Algorithm 1 shields from illegal bases which results in inf or NaN caused by division by zero. The basis expansion step in FFX resembles genetic programming where a large set of bases and their interactions are spanned in search space and randomly selected for evolution. However, in FFX we fill the global search space in one step and later find the optimal models deterministically as discussed in next Section 2.3.

Algorithm 1: Generate interacting variable bases limiting to quadratic nonlinearity

Input: $\widetilde{\mathbf{\Theta}}$ `// input training data`
Output: $\mathbf{\Theta}$ `// list of bases`
$\mathbf{B}_1 = \{\}$

`// input variable` $\widetilde{\mathbf{\Theta}} = \{\widetilde{\theta}_1, \widetilde{\theta}_2, \widetilde{\theta}_3, \ldots\}$
for *i = 1 to length*($\widetilde{\mathbf{\Theta}}$) **do**
  for *j = i to length*($\widetilde{\mathbf{\Theta}}$) **do**
    base $= \widetilde{\theta}_i * \widetilde{\theta}_j$
    if OK(eval(base, $\widetilde{\mathbf{\Theta}}$))
    add base to B_1
  end
end
return $\mathbf{\Theta} = \widetilde{\mathbf{\Theta}} \cup \mathbf{B}_1$

2.3. Model Building

This step forms the core of FFX which concerns with identifying basis functions that may be sparse in nature and simultaneously regressing corresponding coefficients, β_is, using given data or measurements. The coefficient vector, $\boldsymbol{\beta}$, can be computed using the LS minimization. However, LS minimization method is prone to overfitting resulting in high model sensitivity on training data and finding extremely large coefficients to represent the given data. Hence, two regularizing terms, L_1 and L_2 norms on the coefficients $\boldsymbol{\beta}$, are added to the least squares objective function to avoid overfitting and tame the coefficients in finding sparse approximation of $\boldsymbol{\beta}$. The resulting regularized LS objective function is as follows:

$$\beta^* = \arg\min_\beta ||\mathbf{\Theta} \cdot \beta - \mathbf{V}(\mathbf{t})||_2^2 + \lambda_1 ||\beta||_1 + \lambda_2 ||\beta||_2^2 \tag{10}$$

where λ_1 and λ_2 are regularizing weights for L_1 and L_2 penalty functions, respectively.

Regularization in compressive sensing refers to forcing certain characteristics of model-based scientific inference, for example, discouraging complex models or extreme explanations, even if they fit the input data. Solving Equation (10) in FFX is termed as pathwise regularized learning which in compressive sensing is a popular hybrid regularized LS method called elastic net approach [17,18]. Elastic net approach combines popular regularizing LS algorithms, namely LASSO regression [11,14] and ridge (or Tikhonov) regression [16,43]. LASSO and its variants including ridge regression are fundamental to the field of compressive sensing, especially in recovery of sparse signals from random data. LASSO penalizes LS objective function by adding sum of the absolute values (L_1 norm) of the regression coefficients. This forces the shrinkage of certain coefficients simultaneously performing feature elimination as L_1 norm promotes sparsity. In ridge regression, sum of the squares (L_2 norm) of the regression coefficients is added to LS objective function. Ridge regression has effect of grouping several correlated variables with increased stable convergence.

Hence by adding both L_1 norm and L_2 norm to LS objective function, we arrive at elastic net approach with objective function as shown in Equation (10). This hybrid approach helps in finding both sparse and dense models which can be later filtered to find optimum model. The λ_1 and λ_2 in Equation (10) control the amount of sparsity by putting more weight on penalty terms and resulting in shrinkage of coefficients and avoiding overfitting. The λ_1 and λ_2 in Equation (10) are modified by introducing a single variable $\lambda \in [0,1]$ as a mixing parameter and $\alpha \geqslant 0$ called regularizing weight. The resulting modified objective function is shown as:

$$\beta^* = \arg\min_\beta ||\mathbf{\Theta} \cdot \beta - \mathbf{V}(\mathbf{t})||_2^2 + \alpha * \lambda ||\beta||_1 + \alpha(1-\lambda)||\beta||_2^2 \tag{11}$$

where the mixing parameter λ is also called L_1 ratio as $\lambda = \lambda_1$ in Equation (11). Hence, $\lambda = 1$ corresponds to pure LASSO regression and $\lambda = 0$ corresponds to pure ridge regression. The regularizing weight α multiplies the penalty terms and dictates the strength of sparsity needed. Hence, $\alpha = 0$ corresponds to ordinary LS and the larger the α, the greater the number of non-zero terms (sparsity) in the resulting model. The benefit of introducing α is that by varying the strength of α, we can get models with different complexities. Figure 1 is used to illustrate this effect of α on standard diabetic set [44] where increase of α adds more non-zero coefficients to discovered model.

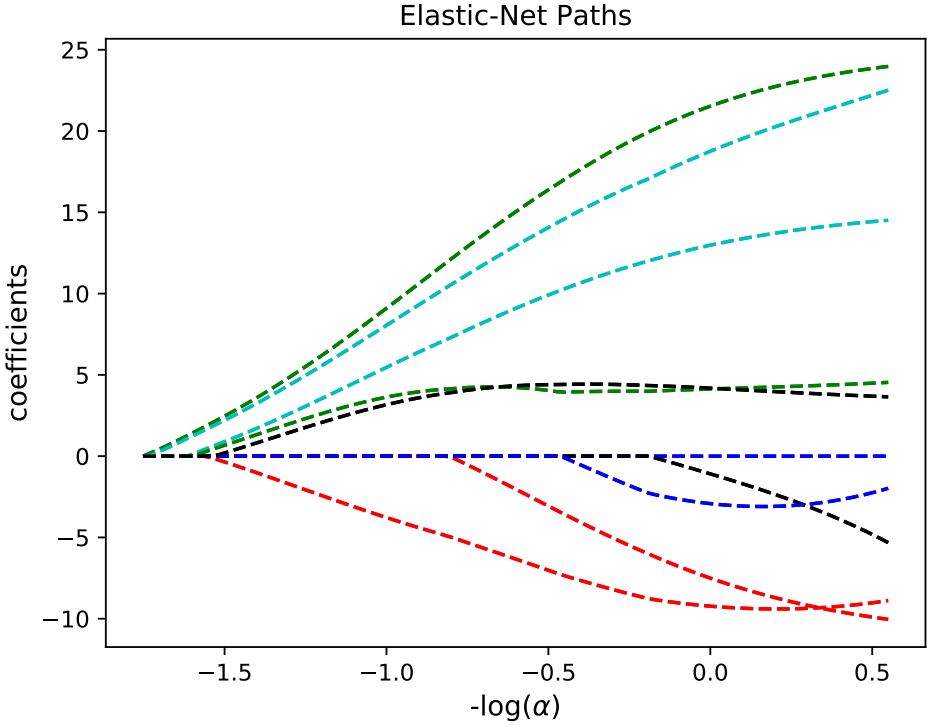

Figure 1. Regularization path for elastic net by varying α on diabetic data set illustrated in [44]. Each line traces the coefficient value of basis functions (features). As α tends to 0, more bases are added to explain data and finally resulting in ordinary LS solution.

In FFX algorithm, elastic net procedure is termed as pathwise regularized learning as it sweeps many values of α automatically by initializing number of αs that need to be searched. Along the path of various αs, Algorithm 2 gives different models with varying complexity and accuracy which are stored for further filtering. The length of the model is restricted through number of maximum bases, which is similar to genetic programming where the maximum length/depth of the tree is restricted as to explain the data with compact model which is easier to interpret. The elastic net method is implemented in FFX using SCIKIT Learn package [44]. We should note that pathwise regularized learning needs to solve quadratic optimization problem. Hence, computation cost increases quadratically with increase in bases N_B.

Algorithm 2: Pathwise learning algorithm

Input: Θ, **V** `// training data, target data`
Output: **Coeff** `// Coefficient vector`

`// Generate range of` α `values:`α_{vec}
$\alpha_{max} = \max(\mathbf{\Theta}^T * \mathbf{V}/(\mathrm{N} * \lambda)$ `//` λ `=` L_1 `ratio.`
`// eps = dictates the length of path.`
eps = 1e-70
$\alpha_{vec} = \mathrm{logspace}(\log_{10}(\alpha_{max} * \mathrm{eps}), \log_{10}(\alpha_{max}, \mathrm{N}_\alpha))$

`// Pathwise Learning()`
$N_{max-bases} = 250$
Coeff = {}
N_{bases} = 0
i = 0
C = zeros(1, length(n))
while $N_B < N_{max-bases}$ *and* $i < length(\alpha_{vec})$ **do**
  $\alpha = \alpha_{vec}$[i]

  `// Using Elastic Net Algorithm from SCIKIT Learn package`
  C = elasticNetLinearFit($\mathbf{\Theta}$, **V**, α, λ, **C**)
  N_B = number of nonzeros values in **C**
  if $N_B < N_{max-bases}$ **then**
    add **C** to **Coeff**
    i = i+1
  end
end
return **Coeff**

2.4. Model Selection

As we sweep through the regularized path by iterating through different regularizing weights, αs, we are storing models with varying complexity (number of bases) to explain the input data. These candidate models are filtered using standard non-dominated sorting to get the Pareto optimum set of best models. A sorting algorithm uses some ranking mechanism to distill best models. For example, NSGA2 [45] uses a heuristic metric called crowding distance to compare two models. This step in FFX is called non-dominated filtering. Algorithm 3 outlines the filtering approach.

The objective of non-dominated filtering is to minimize model complexity (number of bases) along with testing error. As per the Ockham's razor ("law of parsimony"), a simple solution is considered superior to a complicated one [46,47] as it is easier to interpret. In addition, complex functions are prone to overfitting performing too good on training data but worse on unseen (test) data. We look for the set of candidate solutions that are of minimum complexity and test error. The test or training accuracy is calculated by normalized root mean squared error (NRMSE) given by the following expression:

$$\mathrm{NRMSE} = \sqrt{\sum_{i=1}^{n \cdot m} \frac{(\hat{V}(t)_i - V(t)_i)^2}{(max(\boldsymbol{V}) - min(\boldsymbol{V}))^2}} \tag{12}$$

where $\hat{V}(t)$ is the predicted output, $V(t)$ is the actual output and $n \cdot m$ is number of input samples or measurements. Equation (12) is used to determine how well the given model performs on the outside data set which is not shown to the algorithm while training. It is reported in percentage by simply multiplying NRMSE error with 100 and generally called as test error.

Hence, through this filtering, we form Pareto front along which optimal models lie. As this set of solutions satisfy at least minimizing one of the objectives, they are also called non-dominated candidate solutions. These solutions are not worse than any other solutions in the populations compared on all objectives.

We implement FFX setup to recover canonical linear and nonlinear PDEs. We restrict PDEs to be tested up to fifth order in space. The summary of PDEs that are selected for recovery along with their domain and discretization is listed in Table 1. For keeping it simple, we use exact available analytical solutions for PDEs to get the input data for building the library $\widetilde{\Theta}$. Furthermore, all the selected PDEs are non-dimensionalized with respect to length of the domain L and total time T to make the regression process more effective. The resulting non-dimensional forms of selected PDEs are listed in Table 2. Table 3 shows different hyper-parameters values of FFX used in this problem. The L_l ratio, λ, is set at high value (0.95) to promote sparsity in recovered model. Additionally, this value for L_1 ratio (λ) in our setup gives best prediction of coefficients corresponding to basis functions. Future investigations will include more rigorous studies on selecting hyper-parameters using techniques such as cross validation.

To test the robustness of FFX algorithm, we added artificial noise to the analytical solution $u(x,t)$ with different magnitudes of standard deviation of $u(x,t)$. In this framework, noise can be introduced in two ways: (i) first computing the basis functions (i.e., partial derivatives) from recorded data, $u(x,t)$, and then adding noise to each input feature basis function, or (ii) first adding noise to recorded data, $u(x,t)$, and then compute derivatives from noised data and establish a set of basis functions. In this paper, we follow the second approach. We note that this second approach is challenging since the data is noised before we compute the derivatives (candidate basis functions).

Algorithm 3: Non-dominated filter

Input: Coeff, B `// Coefficient vectors, basis functions`
Output: Model `// Pareto optimal Models`

`// Build Models`
$\mathbf{M}_{cand} = \{\}$
for *i = 1 to length(|| Coeff ||)* **do**
  a = **Coeff[i]**
  a_0 = **a[0]**
  a_{nz} = non-zero values in **a**
  $\mathbf{B}_{nz}$ = expressions in **B** corr. to non-zero values in **a**
  m = model(a_0,a_{nz}, B_{nz})
  add m to M_{cand}
end

`// Non-dominated filtering`
$\mathbf{f}_1$ = numBases(m), for each m in M_{cand}
$\mathbf{f}_2$ = testError(m), for each m in M_{cand}
J = nondominatesIndices($\mathbf{f}_1$, $\mathbf{f}_2$)
Model = M_{cand}[j] for each j in J
return **Model**

Table 1. Summary of different canonical PDEs selected for recovery using FFX.

	PDE	Dimensional Equation	Domain	Discretization n (Spatial), m (Temporal)
1.	Wave eq.	$u_t = -au_x$	$x \in [0,1]$ $t \in [0,10]$	$n = 101, m = 2001$
2.	Heat eq.	$u_t = \alpha u_{2x}$	$x \in [-1,1]$ $t \in [0,1]$	$n = 201, m = 101$
3.	Burgers eq.	$u_t = -uu_x + \nu u_{2x}$	$x \in [0,1]$ $t \in [0,100]$	$n = 501, m = 101$
4.	Korteweg–de Vries (KdV) eq.	$u_t = -6uu_x - u_{3x}$	$x \in [-10,10]$ $t \in [0,0.5]$	$n = 501, m = 201$
5.	Kawahara eq.	$u_t = -uu_x - u_{3x} + u_{5x}$	$x \in [-50,50]$ $t \in [0,6]$	$n = 501, m = 751$

Table 2. Non-dimensional form of selected PDEs used to test the FFX.

	PDE	Non-Dimensional Equation	Length Scale (L), Time Scale (T)
1.	Wave eq.	$\widetilde{u}_t = -\frac{T}{L}a\widetilde{u}_x$	$L = 1$ $T = 10$
2.	Heat eq.	$\widetilde{u}_t = \frac{T}{L^2}\alpha\widetilde{u}_{2x}$	$L = 2$ $T = 1$
3.	Burgers eq.	$\widetilde{u}_t = -\widetilde{u}\widetilde{u}_x + \frac{T}{L^2}\nu\widetilde{u}_{2x}$	$L = 1$ $T = 100$
4.	Korteweg–de Vries (KdV) eq.	$\widetilde{u}_t = -6\widetilde{u}\widetilde{u}_x - \frac{T}{L^3}\widetilde{u}_{3x}$	$L = 20$ $T = 0.5$
5.	Kawahara eq.	$\widetilde{u}_t = -\widetilde{u}\widetilde{u}_x - \frac{T}{L^3}\widetilde{u}_{3x} + \frac{T}{L^5}\widetilde{u}_{5x}$	$L = 100$ $T = 6$

Table 3. FFX hyper-parameters used in our setup.

Parameter	Value/Setting
Basis function expansion	Interacting—variable bases
Elastic net	L_1 ratio (λ) $= 0.95$ Number of different α values $= 1000$ Train error $= 1.0 \times 10^{-3}$ Number of maximum bases $= 250$

3. Numerical Experiments

3.1. Wave Equation

Our first test case is wave equation which is a first order linear PDE. The equation takes the form as follows:

$$u_t = -au_x \tag{13}$$

where a is constant wave speed. The traveling single wave solution that satisfies the PDE in Equation (13) is used to get velocity field at various spatial locations and time snapshots which are used as input data for building library of basis functions. The exact analytical solution is given as:

$$u(x,t) = \sin(2\pi(x - at)) \tag{14}$$

where the wave speed is taken as $a = 1$. The solution, Equation (15), has spatial domain $x \in [0, 1]$ and temporal domain $t \in [0, 10]$. The original PDE is non-dimensionalized using length scale L and time scale T. We define dimensionless form

$$\widetilde{u} = \frac{u}{L/T}, \quad \widetilde{x} = \frac{x}{L}, \quad \widetilde{t} = \frac{t}{T}, \tag{15}$$

and omit the tilde from x and t when they refer to the partial derivatives. The resulting non-dimensional PDE has the form:

$$\widetilde{u}_t = -\frac{T}{L} a \widetilde{u}_x \tag{16}$$

where $L = 1$ and $T = 10$ in our set up.

Figure 2 shows the plot of analytical solution for non-dimensional PDE given by Equation (16). Exact solution data is used to form candidate terms consisting of different higher-order partial derivatives. After non-dominated filtering, two models enter into Pareto optimal set as shown in Figure 3 with respect to test error and model complexity (number of bases). Zero complexity in Figure 3 refers to a constant value with high test error. The best model is presented in the colored box in Figure 3. Additionally, for wave equation, FFX does not identify models with intermediate complexity as shown in Figure 3. The recovered model identified by FFX has three bases whereas the wave equation has only one main base term, namely $\widetilde{u}_x$. This is due to FFX adding the terms with corresponding coefficients that are small compared to the main basis function ($\widetilde{u}_x$) to the model. If we look closely, the recovered model can be cleaned by removing the non-significant bases with small values of coefficients and show that the model has only one base term i.e., $\widetilde{u}_x$. This simplification of model to recognize that original PDE being recovered is applied for all numerical experiments carried out in this paper.

Table 4 lists the target and identified PDE recovered from using both clean and noisy data along with their associated test error. As expected, FFX works extremely well using clean data with test error 0.01% compared to their noisy counter parts. This small test error for model identified using clean data is due to the additional terms that are added by FFX to explain the data. As the magnitude of noise is increased, the accuracy of predicted coefficient corresponding to main base term ($\widetilde{u}_x$) drastically decline as shown in Table 4. Finally, at 30% noise, FFX misidentifies the PDE to be recovered.

Table 4. Identifying wave equation using FFX.

Model	Non-Dimensional PDE	Test Error (%)
True	$\widetilde{u}_t = -10.0\ \widetilde{u}_x$	
Recovered (clean)	$\widetilde{u}_t = -4.02 \times 10^{-6} - 10.0\ \widetilde{u}_x + 3.75 \times 10^{-5}\ \widetilde{u}_{3x} - 8.84 \times 10^{-11}\ \widetilde{u}_{5x}$	0.01
Recovered (1% noise)	$\widetilde{u}_t = -0.00255 - 9.91\ \widetilde{u}_x - 0.000344\ \widetilde{u}_{3x}$	5.36
Recovered (5% noise)	$\widetilde{u}_t = -0.0110 - 9.25\ \widetilde{u}_x - 0.000342\ \widetilde{u}_{3x}$	11.85
Recovered (10% noise)	$\widetilde{u}_t = -0.0237 - 7.72\ \widetilde{u}_x - 0.000288\ \widetilde{u}_{3x}$	13.74
Recovered (15% noise)	$\widetilde{u}_t = -0.0384 - 6.13\ \widetilde{u}_x - 0.000228\ \widetilde{u}_{3x}$	14.49
Recovered (20% noise)	$\widetilde{u}_t = -0.0552 - 4.71\ \widetilde{u}_x - 0.000173\ \widetilde{u}_{3x}$	14.89
Recovered (25% noise)	$\widetilde{u}_t = -0.0726 - 3.62\ \widetilde{u}_x - 0.000131\ \widetilde{u}_{3x}$	15.07
Recovered (30% noise)	-0.128	–

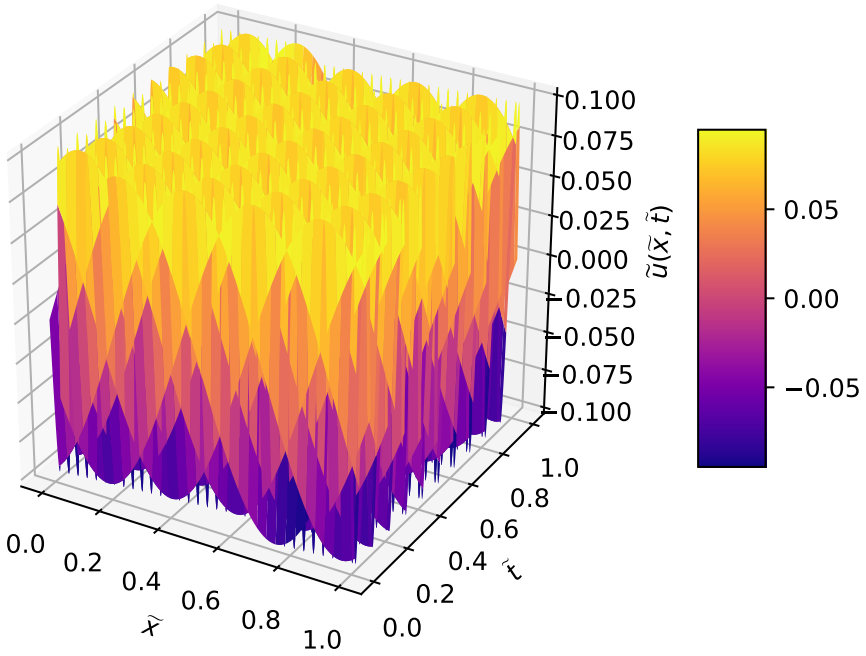

Figure 2. Non-dimensional analytical solution of wave equation.

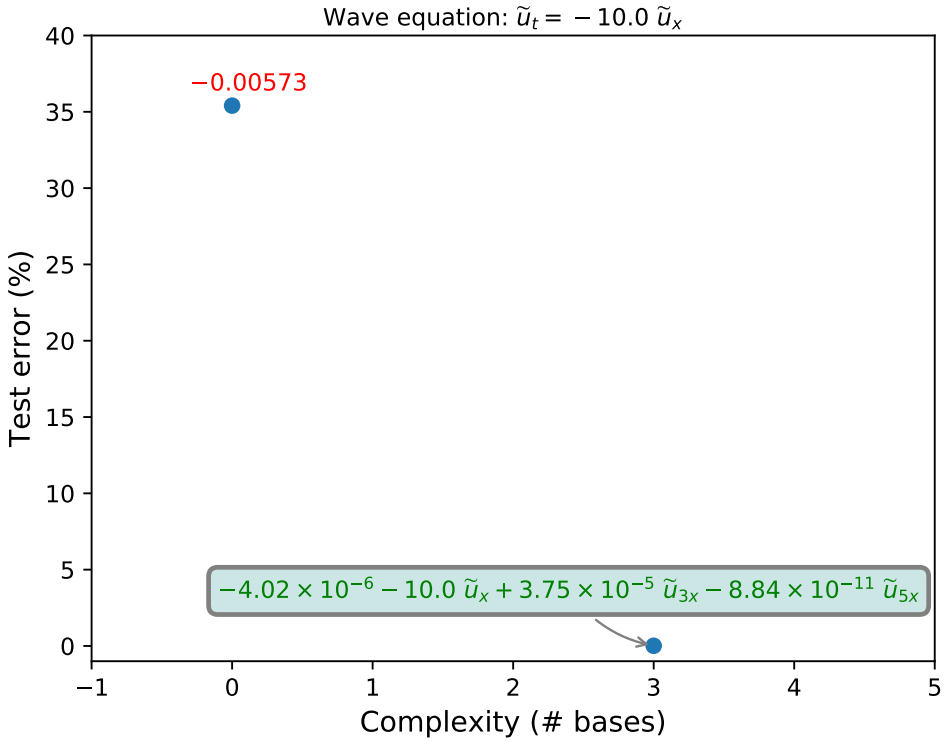

Figure 3. Predicted models with varied test error and complexity for wave equation.

3.2. Heat Equation

We test heat equation which is a second order linear PDE. Heat equation takes the form:

$$u_t = \alpha u_{2x} \tag{17}$$

where the thermal conductivity is taken as $\alpha = \frac{1}{\pi^2}$. Exact solution used in the current paper is given by:

$$u(x,t) = -\sin(\pi x)e^{-t} \tag{18}$$

where the spatial domain $x \in [-1,1]$ and temporal domain $t \in [0,1]$. We nondimensionalize the PDE given by Equation (17) using domain length L and total time T. Resulting non-dimensional PDE is of the form:

$$\widetilde{u}_t = \frac{T}{L^2}\alpha \widetilde{u}_{2x} \tag{19}$$

where $L = 2$ and $T = 1$ in our set up.

Non-dimensional analytical solution to Equation (19) is plotted in Figure 4. Figure 5 shows the Pareto optimal set of models with best model labeled in the colored box. This best model can be cleaned by removing non-significant bases with coefficients vary small compared to the main base term i.e., $\widetilde{u}_{2x}$. Table 5 shows the recovered PDE using both clean and noise data along with associated test error. The small test error for predicted model using clean data in Table 5 is due to addition of these extra basis functions by FFX to explain the data. Recovering heat equation in our set up but FFX is less robust compared to wave equation. Wave equation is recovered by FFX up to 25% noise whereas heat equation is misidentified at 5% noise even though heat equation being linear PDE.

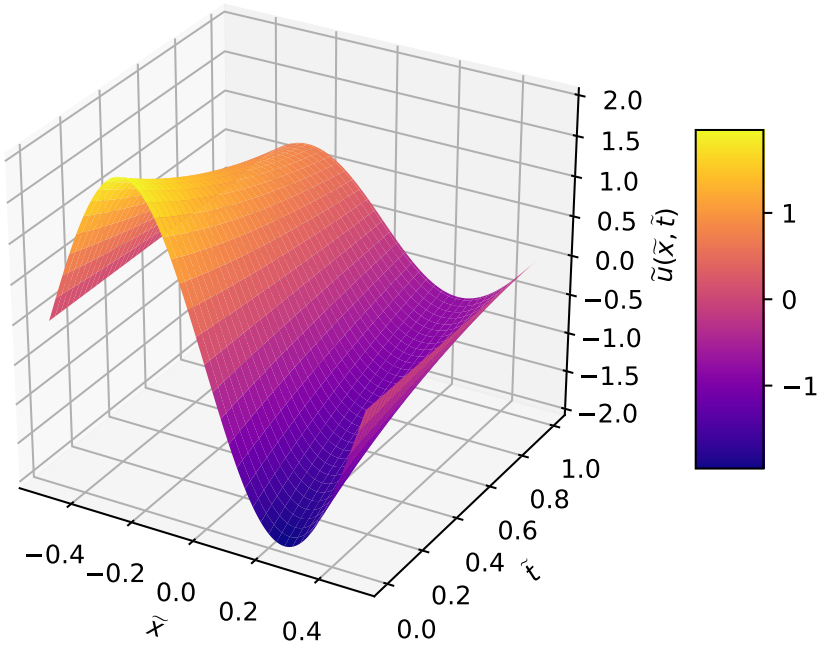

Figure 4. Non-dimensional analytical solution of heat equation.

Table 5. Identifying heat equation using FFX.

Model	Non-Dimensional PDE	Test Error (%)
True	$\widetilde{u}_t = 0.02533\ \widetilde{u}_{2x}$	
Recovered (clean)	$\widetilde{u}_t = 1.33 \times 10^{-5} + 0.0253\ \widetilde{u}_{2x} - 1.25 \times 10^{-7}\ \widetilde{u}_{4x}$	0.01
Recovered (1% noise)	$\widetilde{u}_t = 0.00327 + 0.0269\ \widetilde{u}_{2x} + 6.18 \times 10^{-5}\ \widetilde{u}^2\widetilde{u}_x + 1.40 \times 10^{-7}\ \widetilde{u}_{4x}$	5.16
Recovered (5% noise)	$\widetilde{u}_t = 0.00930$	–

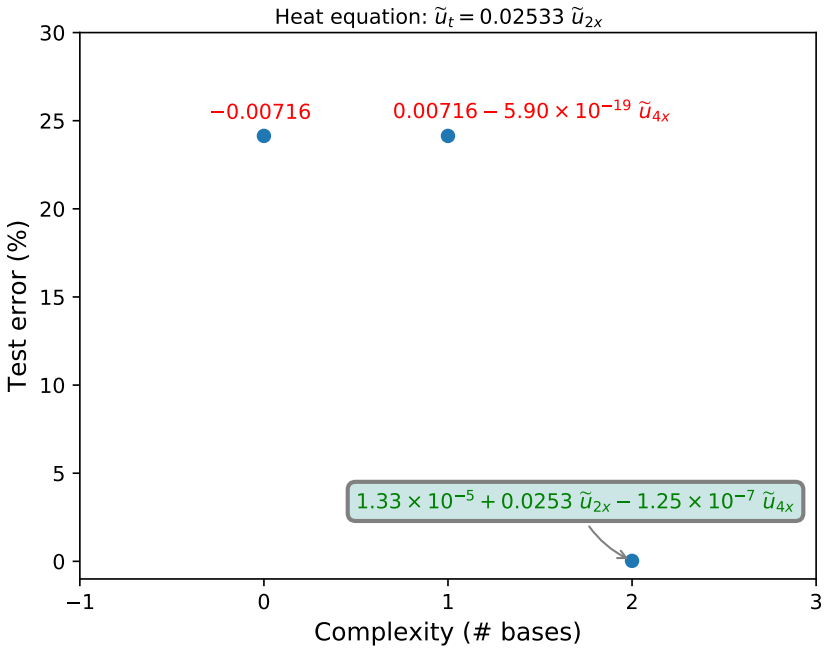

Figure 5. Predicted models with varied test error and complexity for heat equation.

3.3. Burgers Equation

Burgers equation is a fundamental PDE occurring in various areas such as fluid mechanics, nonlinear acoustics and gas dynamics. The interest in Burgers equation arises due to nonlinear term uu_x and present a challenge to FFX algorithm to recover the dynamics. Burgers equation takes the form:

$$u_t = uu_x + \nu u_{2x} \tag{20}$$

where ν is viscosity coefficient. We use available analytical equation to Equation (20) to get solution $u(x,t)$ data. The exact solution is given by:

$$u(x,t) = \frac{2\nu\pi \exp(-\pi^2\nu t)\sin(\pi x)}{a + \exp(-\pi^2\nu t)\cos(\pi x)} \tag{21}$$

where a $= 1.25$ and $\nu = 0.01$. The spatial domain $x \in [0,1]$ and temporal domain $t \in [0,100]$ are taken in Equation (21). The PDE in Equation (20) is nondimensionalized using domain length L and total time T. The resulting non-dimensional PDE is given by:

$$\widetilde{u}_t = -\widetilde{u}\widetilde{u}_x + \frac{T}{L^2}\nu\widetilde{u}_{2x} \tag{22}$$

where $L = 1$ and $T = 100$ in our set up.

Non-dimensional analytical solution is plotted in Figure 6. Figure 7 shows the Pareto optimal set of models with best model highlighted in colored box. This best model can be cleaned by removing non-significant bases with corresponding coefficients close to zero to recover Burgers PDE. Table 6 shows the recovered PDE using both clean and noise data along with the test error. FFX shows promise in recovering nonlinear PDEs as we can see from this current numerical simulation. FFX misidentifies Burgers PDE at 5% noise level as shown in Table 6.

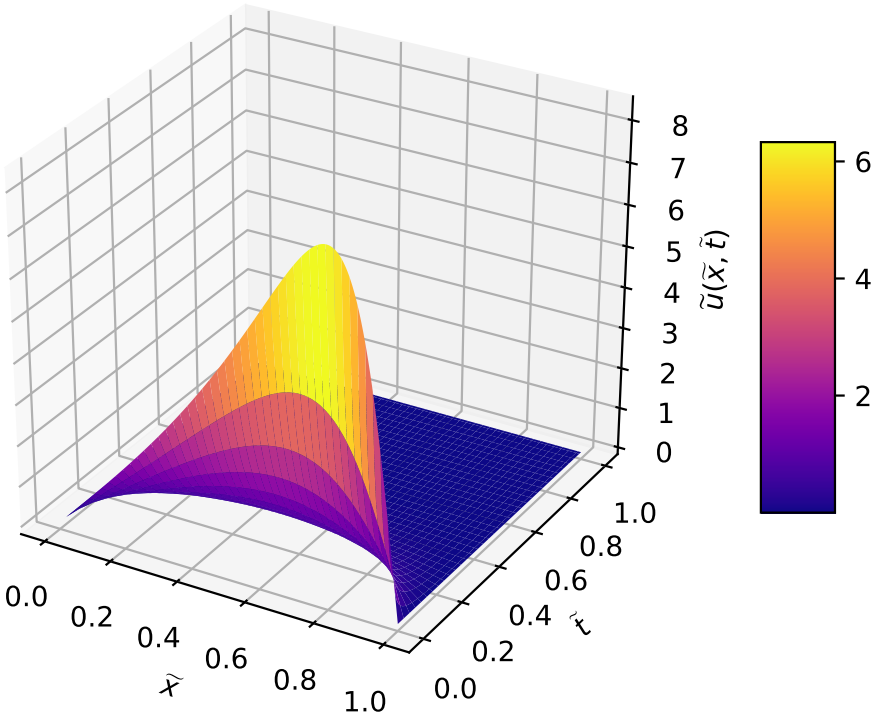

Figure 6. Non-dimensional analytical solution of Burgers equation.

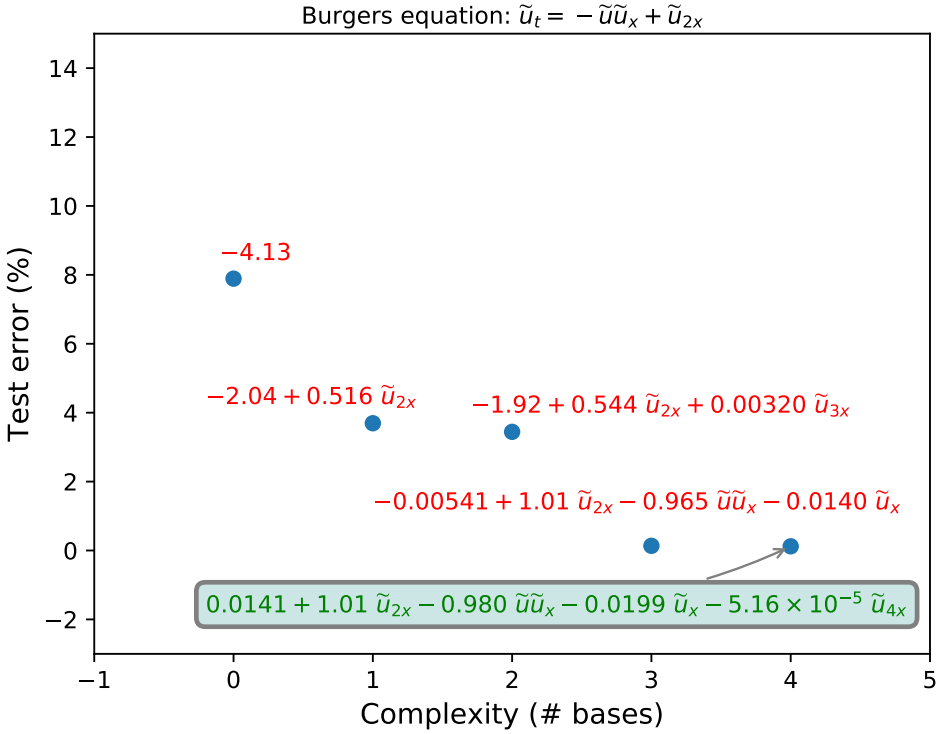

Figure 7. Predicted models with varied test error and complexity for Burgers equation.

Table 6. Identifying Burgers equation using FFX.

Model	Non-Dimensional PDE	Test Error (%)
True	$\widetilde{u}_t = -\widetilde{u}\widetilde{u}_x + \widetilde{u}_{2x}$	
Recovered (clean)	$\widetilde{u}_t = 0.0141 + 1.01\ \widetilde{u}_{2x} - 0.980\ \widetilde{u}\widetilde{u}_x - 0.0199\ \widetilde{u}_x - 5.16 \times 10^{-5}\ \widetilde{u}_{4x}$	0.12
Recovered (1% noise)	$\widetilde{u}_t = 0.0986 + 1.21\ \widetilde{u}_{2x} - 0.957\ \widetilde{u}\widetilde{u}_x - 0.00144\ \widetilde{u}_x + 5.41 \times 10^{-7}\ \widetilde{u}_{4x}$	5.87
Recovered (5% noise)	$\widetilde{u}_t = -4.14 + 0.36\ \widetilde{u}\widetilde{u}_x$	–

3.4. Korteweg–de Vries Equation

Korteweg–de Vries (KdV) equation is used to model the behavior of magneto hydrodynamics waves in warm plasma, acoustic waves in an inharmonic crystal and ion-acoustic waves [48]. Many different forms KdV equations exist but we use the form as given below:

$$u_t = -6\ uu_x - u_{3x} \tag{23}$$

We use analytical equation for KdV equation (Equation (23)) given in [49] which is an interacting wave solution. The solution with multiple soliton interactions exhibit highly nonlinear behavior which challenges FFX apart from being higher-order PDE. The exact equation is given as:

$$u(x,t) = 12\left(\frac{4\cosh(2x-8t) + \cosh(4x-64t) + 3}{(3\cosh(x-28t) + \cosh(3x-36t))^2}\right) \tag{24}$$

where the spatial domain $x \in [-10, 10]$ and temporal domain $t \in [0, 0.5]$ are taken for solving Equation (24). The two solitons interacting with each other is shown in the Figure 8. The resulting non-dimensional PDE with respect to domain length L and total time T is shown below:

$$\widetilde{u}_t = -6\ \widetilde{u}\widetilde{u}_x - \frac{T}{L^3}\ \widetilde{u}_{3x} \tag{25}$$

where $L = 20$ and $T = 0.5$ in our setup.

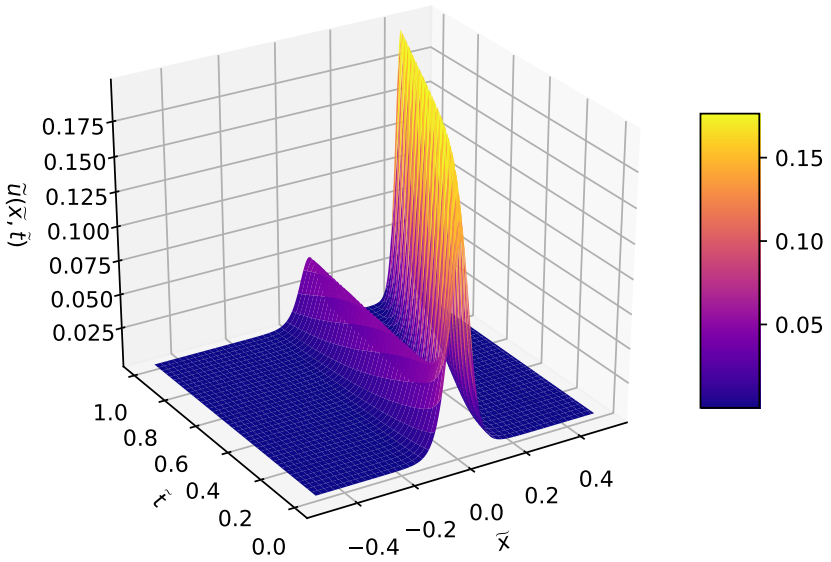

Figure 8. Non-dimensional analytical solution of KdV equation.

Pareto optimal set with best model in colored box is plotted in Figure 9. This best model can be cleaned by removing non-significant bases with corresponding coefficients very small compared to main basis functions, namely $\widetilde{u}\widetilde{u}_x$ and $\widetilde{u}_{3x}$ to recover KdV equation. FFX recovers the original

non-dimensional PDE for both clean and noise cases as shown in Table 7. This test case shows FFX is reliable for recovering dynamics governed by complex multiple interacting waves. Furthermore, KdV equation is identified by FFX at 1% noise level but fails to recover at 5% noise.

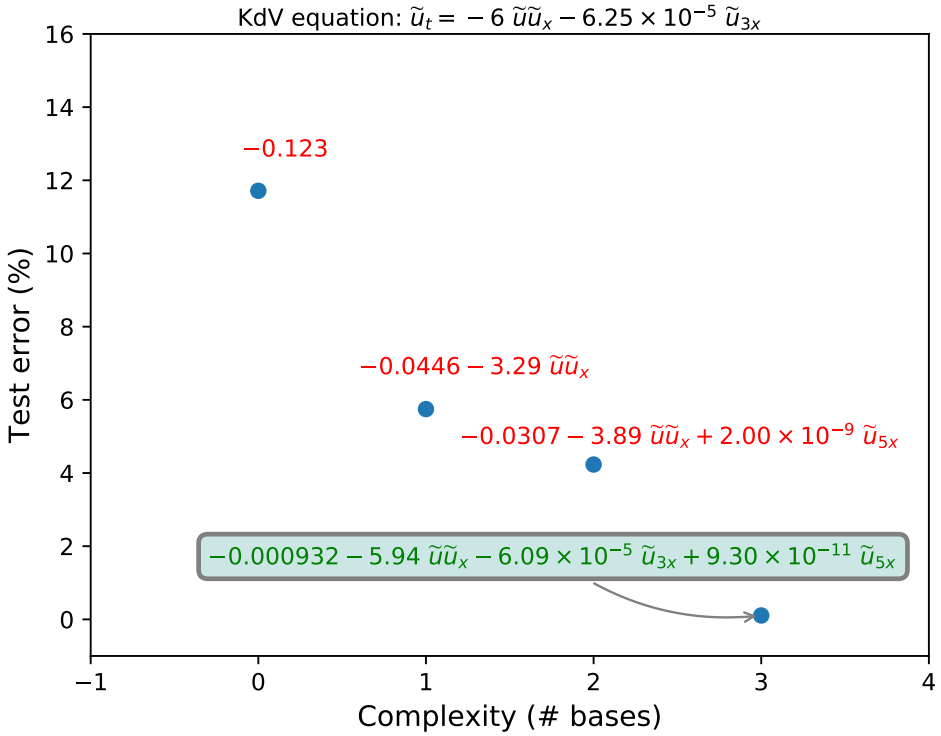

Figure 9. Predicted models with varied test error and complexity for KdV equation.

Table 7. Identifying KdV equation using FFX.

Model	Non-Dimensional PDE	Test Error (%)
True	$\widetilde{u}_t = -6\,\widetilde{u}\widetilde{u}_x - 6.25 \times 10^{-5}\,\widetilde{u}_{3x}$	
Recovered (clean)	$\widetilde{u}_t = -0.000932 - 5.94\,\widetilde{u}\widetilde{u}_x - 6.09 \times 10^{-5}\,\widetilde{u}_{3x} + 9.30 \times 10^{-11}\,\widetilde{u}_{5x}$	0.11
Recovered (1% noise)	$\widetilde{u}_t = 0.0825 - 5.86\,\widetilde{u}\widetilde{u}_x - 5.75 \times 10^{-5}\,\widetilde{u}_{3x} - 6.925 \times 10^{-7}\,\widetilde{u}^2\widetilde{u}_{4x}$	6.75
Recovered (5% noise)	$\widetilde{u}_t = -0.0430 - 7.64\,\widetilde{u}\widetilde{u}_x + 0.164\,\widetilde{u}_x - 0.000235\,\widetilde{u}_{2x} - 2.16 \times 10^{-7}\,\widetilde{u}_{3x}$	–

3.5. Kawahara Equation

We consider a fifth-order nonlinear PDE called Kawahara equation introduced in [50]. This equation is also known as a fifth-order KdV equation or singularly perturbed KdV equation. The fifth-order KdV equation is one of the most known nonlinear evolution equation which is used in the theory of magneto-acoustic waves in a plasma [50], capillary-gravity waves [51] and in the theory of shallow water waves [52]. The Kawahara equation takes the form:

$$u_t = -uu_x - u_{3x} + u_{5x} \tag{26}$$

We use single traveling solitary solution for Equation (26) to get the solution field $u(x,t)$. The exact wave form solution listed below is taken from [53]:

$$u(x,t) = \frac{105}{169}\,\mathrm{sech}\left(\frac{1}{2\sqrt{13}}\,(x - at)\right)^4 \tag{27}$$

where the wave travels with the speed $a = \frac{36}{169}$, the spatial domain $x \in [-50, 50]$ and temporal domain $t \in [0, 6]$. The Equation (26) is nondimensionalized using domain length L and total time T resulting in the form as shown below:

$$\widetilde{u}_t = -\widetilde{u}\widetilde{u}_x - \frac{T}{L^3}\widetilde{u}_{3x} + \frac{T}{L^5}\widetilde{u}_{5x} \tag{28}$$

where $L = 100$ and $T = 6$ in our setup. The non-dimensional solution is plotted in Figure 10 show the single wave soliton clearly.

This test case is interesting as the analytical solution also satisfies the wave equation with wave speed, $a = \frac{36}{169}$. Pathwise regularized learning (elastic net approach) finds the models that are sparser in nature. Hence FFX converges to wave equation with speed a. This is proved in Figure 11 which shows the wave equation in colored box being recovered for the same solution given by Equation (27).

The ambiguity in identifying intended models with input data satisfying multiple equations is a concern and actively being pursued for future research. In the current paper, to test the performance of FFX to recover higher-order PDEs, we remove the basis function u_x from the library $\boldsymbol{\Theta}$ to discourage FFX to converge to wave equation. Figure 12 shows the Pareto front of optimal model with best model highlighted in colored box. Table 8 shows the recovery of PDE for both clean and noise data. This workaround demonstrates the FFX algorithm to recover higher-order PDEs such as Kawahara equation. Furthermore, at 5% noise level, FFX misidentifies Kawahara equation just like other nonlinear PDEs tested before.

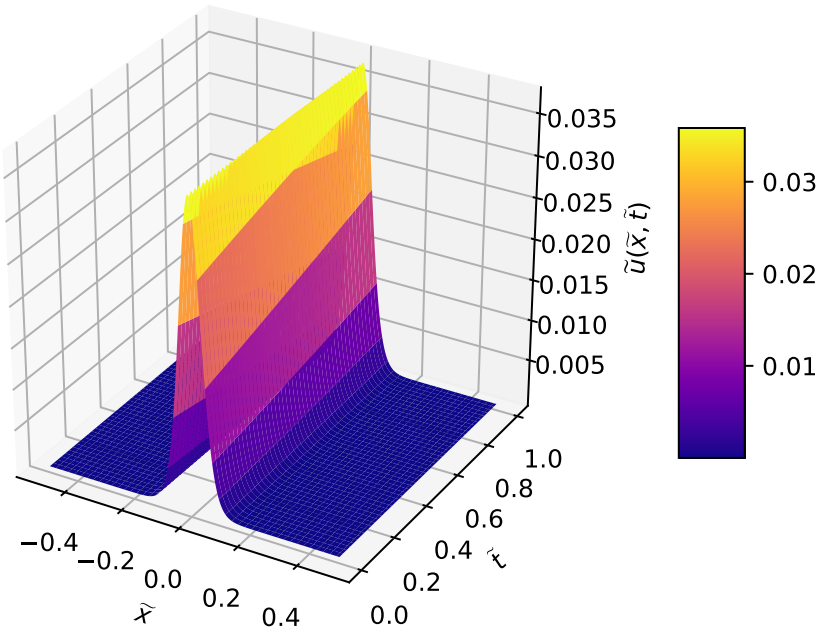

Figure 10. Non-dimensional analytical solution of Kawahara equation.

Table 8. Identifying Kawahara equation using FFX.

Model	Non-Dimensional PDE	Test Error (%)
True	$\widetilde{u}_t = -\widetilde{u}\widetilde{u}_x - 6.0 \times 10^{-6}\ \widetilde{u}_{3x} - 6.0 \times 10^{-10}\ \widetilde{u}_{5x}$	
Recovered (clean)	$-5.26 \times 10^{-9} - 0.999\ \widetilde{u}\widetilde{u}_x - 5.76 \times 10^{-6}\ \widetilde{u}_{3x} - 6.24 \times 10^{-10}\ \widetilde{u}_{5x}$	0.18
Recovered (1% noise)	$\widetilde{u}_t = 0.0008965 - 0.994\ \widetilde{u}\widetilde{u}_x - 5.16 \times 10^{-6}\ \widetilde{u}_{3x} - 6.76 \times 10^{-10}\ \widetilde{u}_{5x}$	6.57
Recovered (5% noise)	$\widetilde{u}_t = -1.85 \times 10^{-5} + 1.51 \times 10^{-5}\ \widetilde{u}^2\widetilde{u}_x - 0.578\ \widetilde{u}_{5x} - 2.47 \times 10^{-7}\ \widetilde{u}_x - 4.31 \times 10^{-9}\ \widetilde{u}_{2x}$	–

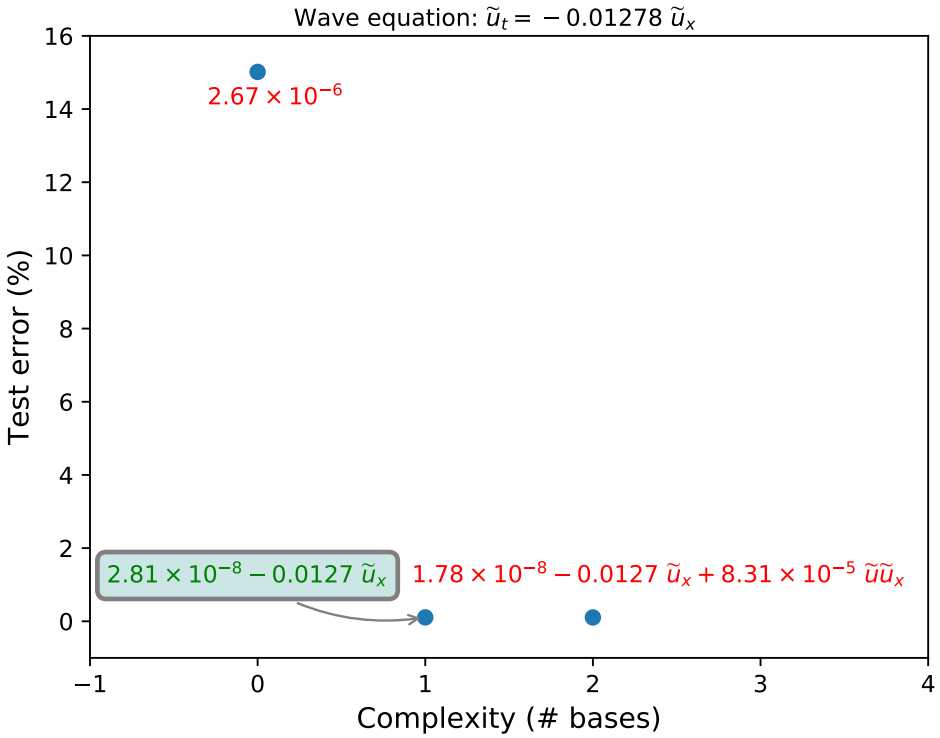

Figure 11. Predicted wave equation rather the intended Kawahara equation as the input data satisfies both PDEs. FFX identifies the sparser (lesser bases) model which is the wave equation.

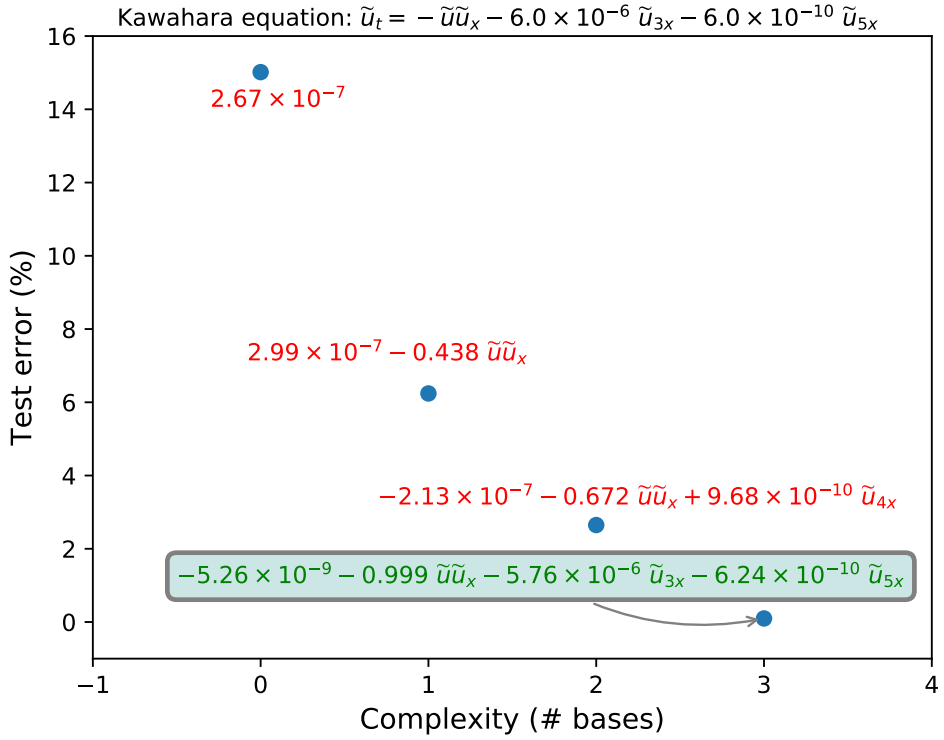

Figure 12. Predicted models with varied test error and complexity for Kawahara equation. Note: the basis u_x is removed from the candidate bases.

4. Summary and Conclusions

Machine-learning methods can be extremely useful for researchers in inferring complex models from data or measurements. FFX mainly leverages recent advances in compressive sensing to learn analytically tractable mathematical models using only data. This makes FFX a deterministic symbolic regression approach. The core of FFX is pathwise regularized learning also called elastic net algorithm, which is a popular sparse regression approach. The sparse regression methods regularize the ordinary least squares regression problem by adding L_1 and L_2 penalty terms to discourage overfitting and tame the coefficients. This regularization of LS problem promotes sparsity there by recovering only a subset of candidate bases to explain the data. This property is what made sparse regression approaches such as LASSO, ridge, and their variants fundamental to compressive sensing. FFX enumerates the given basis functions to add nonlinear terms and uses pathwise regularized learning to extract several models of varying complexity (number of bases) and prediction accuracy. These models are then filtered using non-dominated sorting favoring lower-complexity models with best test accuracy. The enumeration of basis functions to add nonlinear bases to library and non-dominated sorting of learned models with respect to model complexity and test accuracy are inspired by GP. The core of FFX is pathwise regularized learning (elastic net) inspired by CS which is a regularized LS problem.

In this paper, we demonstrate the use of FFX to extract different linear and nonlinear PDEs by exploring patterns in data. We especially build and enumerate large candidate features using input data to leverage sparse optimization algorithm of FFX to discover parsimonious equations. The numerical experiments of several canonical PDEs shows that FFX is a promising machine-learning technique to capture true features and associated coefficients accurately. Additionally, input sensor data is slightly distorted by adding noise to test the robustness of the algorithm (i.e., adding noise before computing candidate basis functions). FFX works reasonably well up to 1% of noise but fails for higher amounts of noise. Additionally, wave equation is recovered for higher levels of noise (up to 25%) compared to other PDEs. This aspect of algorithm needs further investigation as the real-world data sets might have higher levels of noise. However, we note that adding noise to input data which is later used to enumerate features might be more challenging than adding noise directly to candidate basis functions (not shown in this study). Additionally, there is a challenge in identifying the desired model if the solution satisfies multiple equations. This is due to the sparsity-promoting behavior of sparse regression methods which converges to the optimal model containing the least number of bases to explain the data, e.g., it finds wave equations even though the solution satisfies both wave and Kawahara equations. Overall, FFX is a deterministic and scalable algorithm that can be exploited as a potential data-driven tool for recovering hidden physical structures or parameterizations representing high-dimensional systems such as turbulent geophysical flows, traffic models, and weather forecasting using only data.

Author Contributions: Conceptualization, H.V. and O.S.; Data curation, H.V.; Formal analysis, H.V.; Visualization, H.V.; Methodology, H.V. and O.S.; Supervision, O.S.; Writing—original draft, H.V.; and Writing—review and editing, H.V. and O.S.

Funding: This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Number DE-SC0019290. Disclaimer: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Acknowledgments: O.S. gratefully acknowledges the support received by the U.S. Department of Energy via DE-SC0019290.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jordan, M.I.; Mitchell, T.M. Machine learning: Trends, perspectives, and prospects. *Science* **2015**, *349*, 255–260. [CrossRef]
2. Marx, V. Biology: The big challenges of big data. *Nature* **2013**, *498*, 255–260. [CrossRef] [PubMed]
3. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386. [CrossRef] [PubMed]
4. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef] [PubMed]
5. Ciregan, D.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3642–3649.
6. Karpathy, A.; Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3128–3137.
7. Sallab, A.E.; Abdou, M.; Perot, E.; Yogamani, S. Deep reinforcement learning framework for autonomous driving. *Electron. Imaging* **2017**, *2017*, 70–76. [CrossRef]
8. Al Mamun, S.; Lu, C.; Jayaraman, B. Extreme learning machines as encoders for sparse reconstruction. *Fluids* **2018**, *3*, 88. [CrossRef]
9. Donoho, D.L. Compressed sensing. *IEEE Trans. Inf. Theory* **2006**, *52*, 1289–1306. [CrossRef]
10. Candes, E.J.; Wakin, M.B. An introduction to compressive sampling. *IEEE Signal Process. Mag.* **2008**, *25*, 21–30. [CrossRef]
11. Tibshirani, R. Regression shrinkage and selection via the LASSO. *J. R. Stat. Soc. Ser. B* **1996**, *58*, 267–288. [CrossRef]
12. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer Science + Business Media: New York, NY, USA, 2013; Volume 112.
13. Rauhut, H. Compressive sensing and structured random matrices. *Theor. Found. Numer. Methods Sparse Recovery* **2010**, *9*, 1–92.
14. Tibshirani, R.; Wainwright, M.; Hastie, T. *Statistical Learning with Sparsity: The LASSO and Generalizations*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2015.
15. Candes, E.J.; Romberg, J.K.; Tao, T. Stable signal recovery from incomplete and inaccurate measurements. *Commun. Pure Appl. Math. J. Issued Courant Inst. Math. Sci.* **2006**, *59*, 1207–1223. [CrossRef]
16. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
17. Zou, H.; Hastie, T. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B* **2005**, *67*, 301–320. [CrossRef]
18. Friedman, J.; Hastie, T.; Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.* **2010**, *33*, 1. [CrossRef] [PubMed]
19. Brunton, S.L.; Proctor, J.L.; Kutz, J.N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. USA* **2016**, *113*, 3932–3937. [CrossRef] [PubMed]
20. Rudy, S.H.; Brunton, S.L.; Proctor, J.L.; Kutz, J.N. Data-driven discovery of partial differential equations. *Sci. Adv.* **2017**, *3*, e1602614. [CrossRef] [PubMed]
21. Schaeffer, H.; Caflisch, R.; Hauck, C.D.; Osher, S. Sparse dynamics for partial differential equations. *Proc. Natl. Acad. Sci. USA* **2013**, *110*, 6634–6639. [CrossRef]
22. Schaeffer, H. Learning partial differential equations via data discovery and sparse optimization. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2017**, *473*, 20160446. [CrossRef]
23. Tran, G.; Ward, R. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Model. Simul.* **2017**, *15*, 1108–1129. [CrossRef]
24. Schaeffer, H.; Tran, G.; Ward, R. Extracting sparse high-dimensional dynamics from limited data. *SIAM J. Appl. Math.* **2018**, *78*, 3279–3295. [CrossRef]
25. Mangan, N.M.; Kutz, J.N.; Brunton, S.L.; Proctor, J.L. Model selection for dynamical systems via sparse regression and information criteria. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2017**, *473*, 20170009. [CrossRef]
26. Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; MIT Press: Cambridge, MA, USA, 1992; Volume 1.

27. Ferreira, C. *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 21.
28. Schmidt, M.; Lipson, H. Distilling free-form natural laws from experimental data. *Science* **2009**, *324*, 81–85. [CrossRef] [PubMed]
29. Rodriguez-Vazquez, K.; Fleming, P.J. Multi-objective genetic programming for nonlinear system identification. *Electron. Lett.* **1998**, *34*, 930–931. [CrossRef]
30. Yang, Y.; Wang, C.; Soh, C. Force identification of dynamic systems using genetic programming. *Int. J. Numer. Methods Eng.* **2005**, *63*, 1288–1312. [CrossRef]
31. Ferariu, L.; Patelli, A. Multiobjective genetic programming for nonlinear system identification. In Proceedings of the International Conference on Adaptive and Natural Computing Algorithms, Kuopio, Finland, 23–25 April 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 233–242.
32. Brunton, S.L.; Noack, B.R. Closed-loop turbulence control: Progress and challenges. *Appl. Mech. Rev.* **2015**, *67*, 050801. [CrossRef]
33. Gautier, N.; Aider, J.L.; Duriez, T.; Noack, B.; Segond, M.; Abel, M. Closed-loop separation control using machine learning. *J. Fluid Mech.* **2015**, *770*, 442–457. [CrossRef]
34. Luo, C.; Zhang, S.L. Parse-matrix evolution for symbolic regression. *Eng. Appl. Artif. Intell.* **2012**, *25*, 1182–1193. [CrossRef]
35. Brameier, M.F.; Banzhaf, W. *Linear Genetic Programming*; Springe: New York, NY, USA, 2007.
36. Weatheritt, J.; Sandberg, R. A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship. *J. Comput. Phys.* **2016**, *325*, 22–37. [CrossRef]
37. Schoepplein, M.; Weatheritt, J.; Sandberg, R.; Talei, M.; Klein, M. Application of an evolutionary algorithm to LES modelling of turbulent transport in premixed flames. *J. Comput. Phys.* **2018**, *374*, 1166–1179. [CrossRef]
38. McConaghy, T. FFX: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice IX*; Springer: New York, NY, USA, 2011; pp. 235–260.
39. Nelder, J.A.; Wedderburn, R.W. Generalized linear models. *J. R. Stat. Soc. Ser. A* **1972**, *135*, 370–384. [CrossRef]
40. Quade, M.; Abel, M.; Shafi, K.; Niven, R.K.; Noack, B.R. Prediction of dynamical systems by symbolic regression. *Phys. Rev. E* **2016**, *94*, 012214. [CrossRef]
41. Chen, C.; Luo, C.; Jiang, Z. Elite bases regression: A real-time algorithm for symbolic regression. In Proceedings of the 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 29–31 July 2017; pp. 529–535.
42. Worm, T.; Chiu, K. Prioritized grammar enumeration: Symbolic regression by dynamic programming. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 1021–1028.
43. Ng, A.Y. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In Proceedings of the Twenty-First International Conference on Machine Learning, Banff, AB, Canada, 4–8 July 2004; p. 78.
44. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
45. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
46. Blumer, A.; Ehrenfeucht, A.; Haussler, D.; Warmuth, M.K. Occam's razor. *Inf. Process. Lett.* **1987**, *24*, 377–380. [CrossRef]
47. Berger, J.O.; Jefferys, W.H. The application of robust Bayesian analysis to hypothesis testing and Occam's razor. *J. Ital. Stat. Soc.* **1992**, *1*, 17–32. [CrossRef]
48. Ozis, T.; Ozer, S. A simple similarity-transformation-iterative scheme applied to Korteweg–de Vries equation. *Appl. Math. Comput.* **2006**, *173*, 19–32.
49. Lamb, G.L., Jr. *Elements of Soliton Theory*; Wiley-Interscience: New York, NY, USA, 1980.
50. Kawahara, T. Oscillatory solitary waves in dispersive media. *J. Phys. Soc. Jpn.* **1972**, *33*, 260–264. [CrossRef]

51. Kawahara, T.; Sugimoto, N.; Kakutani, T. Nonlinear interaction between short and long capillary-gravity waves. *J. Phys. Soc. Jpn.* **1975**, *39*, 1379–1386. [CrossRef]
52. Hunter, J.K.; Scheurle, J. Existence of perturbed solitary wave solutions to a model equation for water waves. *Phys. D Nonlinear Phenom.* **1988**, *32*, 253–268. [CrossRef]
53. Sirendaoreji. New exact travelling wave solutions for the Kawahara and modified Kawahara equations. *Chaos Solitons Fract.* **2004**, *19*, 147–150. [CrossRef]