Data-driven discovery of time fractional differential equations[⋆]

Abhishek Kumar Singh[1][0000−0002−1263−687X], Mani Mehra[1][0000−0002−4261−7784], and Anatoly A. Alikhanov[2][0000−0003−0684−6667]

[1] Indian Institute of Technology Delhi, New Delhi-110016, India
[2] North-Caucasus Center for Mathematical Research, North-Caucasus Federal University
Stavropol-355017, Russia
assinghabhi@gmail.com
mmehra@maths.iitd.ac.in
aaalikhanov@gmail.com

Abstract. In the era of data abundance and machine learning technologies, we often encounter difficulties in learning data-driven discovery of hidden physics, that is, learning differential equations/fractional differential equations via data. In [1], Schaeffer proposed a machine learning algorithm to learn the differential equation via data discovery. We extend Schaeffer's work in the case of time fractional differential equations and propose an algorithm to identify the fractional order α and discover the form of $\mathcal{F}$. Furthermore, if we have prior information regarding the set in which parameters belong to have some advantages in terms of time complexity of the algorithm over Schaeffer's work. Finally, we conduct various numerical experiments to verify the method's robustness at different noise levels.

Keywords: Fractional differential equations · Sparse optimization · Machine learning · Differential evolution.

1 Introduction

The differential equations describe many phenomena in science and engineering, such as the Schrödinger equation, diffusion equation, a system of differential equations (SIR epidemic model), etc (see [2] and references therein). The original discovery of these equations require a tremendous knowledge of physics, understanding of theories, and supportive evidence of experimental data. This is the one way to discover hidden physics. In recent years, researchers have provided the computational approach to data-driven discovery of hidden physics, that is, learning the differential equations that govern a particular phenomenon only using the data. Machine learning offers a wide range of numerical tools that efficiently learn differential equations via data discovery. Researchers in many fields have applied these tools to discover physical law from experimental data. Raissi *et al.* [3] used probabilistic machine learning to discover the linear differential equations. Again, Raissi

[⋆] Supported by University Grants Commission India, SERB India and North-Caucasus Center for Mathematical Research, Russia.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_8

et al. [4] introduced a physics-informed neural network to discover the non-linear partial differential equations. Due to some issues with the regular data training in the physics-informed neural network, Krishanpriyan *et al.* [5] introduced a physics-informed neural network with some advanced training techniques to discover non-linear partial differential equations. In the works mentioned above, authors know the form of the differential equation, whether its diffusion or diffusion-reaction, etc. The first time, Schaeffer [1] introduced machine learning techniques to identify the terms in underlying partial differential equations to the best of our knowledge. However, in [6], Srivastava *et al.* proposed a machine learning approach to identify the terms in underlying partial differential equations. In recent decades, the theory of fractional derivatives has been an emerging topic in physical, life, and social sciences due to its non-local properties. The non-local properties of fractional operators give a superior way to deal with the complex phenomena in physical, life and social sciences (see [7] and references therein). Gulian *et al.* [8] extend the Raissi *et al.* [3] work to find the space fractional differential equations. Pang *et al.* [9] generalized the Raissi *et al.* [4] work to learn the space-time fractional differential equations. Recently, Singh *et al.* [10, 11] proposed scientific machine learning algorithm to learn the system of fractional differential equations via data.

Motivated by the above discussions, we extend the Schaeffer [1] work for time fractional differential equations. The primary approach used by Schaeffer is to convert the problem into an optimization problem and then use the indirect method to solve the optimization problem. Schaeffer used the least absolute shrinkage and selection operator (LASSO) method, which uses the L_1-norm in the regularization term, with the help of a linear system of equations [12]. In this work, we also use the LASSO method with the help of a non-linear system of equations and use differential evolution to solve the LASSO method.

2 Methodology

In this section, we generalized the methodology, recently given by Schaeffer [1], in the case of time fractional differential equations.

2.1 Sparse reconstruction of Fractional Differential Equation

We will derive the method for a fractional differential equation which has the following form:

$$ {}_{\mathrm{C}}D^{\alpha}_{0,t}\mathcal{U}(x,t) = \mathcal{F}(\mathcal{U}, \mathcal{U}_x, \mathcal{U}_{xx}, \cdots, \mathcal{U}_{x\cdots x}) + \mathcal{G}(x,t), \tag{1} $$

where $0 < \alpha < 1$, ${}_{\mathrm{C}}D^{\alpha}_{0,t}$ denote the left-side Caputo fractional derivative of order α with respect to t (for definition of Caputo see the [11]). Assume that the form of the $\mathcal{F}$ and fractional order α in the above Equation are unknown to the user, and instead user only have the data. For understanding and derivation of the methodology, here we consider the following form:

$$ {}_{\mathrm{C}}D^{\alpha}_{0,t}\mathcal{U}(x,t) - \mathcal{G}(x,t) = \mathcal{F}(\mathcal{U}, \mathcal{U}_x, \mathcal{U}_{xx}). \tag{2} $$

We can approximate the function $\mathcal{F}$ via a n^{th}-order Taylor series expansion. This approximation can be made by the user based on some physically relevant models related to the data. Here for simplicity we take $n = 2$, the second order Taylor series expansion of $\mathcal{F}$ about the origin can be written as:

$$\begin{aligned}\mathcal{F}(\mathcal{U},\mathcal{U}_x,\mathcal{U}_{xx}) &\approx \mathcal{F}(0,0,0) + \mathcal{U}\frac{\partial \mathcal{F}}{\partial \mathcal{U}}(0,0,0) + \mathcal{U}_x\frac{\partial \mathcal{F}}{\partial \mathcal{U}_x}(0,0,0) + \mathcal{U}_{xx}\frac{\partial \mathcal{F}}{\partial \mathcal{U}_{xx}}(0,0,0) \\ &+ \frac{1}{2}\left[\mathcal{U}^2\frac{\partial^2 \mathcal{F}}{\partial \mathcal{U}^2}(0,0,0) + \mathcal{U}_x^2\frac{\partial^2 \mathcal{F}}{\partial \mathcal{U}_x^2}(0,0,0) + \mathcal{U}_{xx}^2\frac{\partial^2 \mathcal{F}}{\partial \mathcal{U}_{xx}^2}(0,0,0)\right. \\ &\left. + 2\mathcal{U}\mathcal{U}_x\frac{\partial^2 \mathcal{F}}{\partial \mathcal{U}\partial \mathcal{U}_x}(0,0,0) + 2\mathcal{U}\mathcal{U}_{xx}\frac{\partial^2 \mathcal{F}}{\partial \mathcal{U}\partial \mathcal{U}_{xx}}(0,0,0) + 2\mathcal{U}_x\mathcal{U}_{xx}\frac{\partial^2 \mathcal{F}}{\partial \mathcal{U}_x\partial \mathcal{U}_{xx}}(0,0,0)\right].\end{aligned} \tag{3}$$

Using the Equation (3) in the Equation (2), we can write

$${}_{\mathrm{C}}D_{0,t}^{\alpha}\mathcal{U}(x,t) - \mathcal{G}(x,t) = \alpha_1 + \mathcal{U}\alpha_2 + \mathcal{U}_x\alpha_3 + \cdots + \mathcal{U}_x\mathcal{U}_{xx}\alpha_{10}, \tag{4}$$

where $\alpha_1 = \mathcal{F}(0,0,0), \alpha_2 = \frac{\partial \mathcal{F}}{\partial \mathcal{U}}(0,0,0), \alpha_3 = \frac{\partial \mathcal{F}}{\partial \mathcal{U}_x}(0,0,0), \cdots, \alpha_{10} = \frac{\partial^2 \mathcal{F}}{\partial \mathcal{U}_x \partial \mathcal{U}_{xx}}(0,0,0)$ are the coefficients of the unknown in the Equation (3). We can write the above equation as:

$${}_{\mathrm{C}}D_{0,t}^{\alpha}\mathcal{U}(x,t) - \mathcal{G}(x,t) = [1\ \mathcal{U}\ \mathcal{U}^2\ \mathcal{U}_x\ \mathcal{U}_x^2\ \mathcal{U}\mathcal{U}_x\ \mathcal{U}_{xx}\ \mathcal{U}_{xx}^2\ \mathcal{U}\mathcal{U}_{xx}\ \mathcal{U}_x\mathcal{U}_{xx}].\boldsymbol{\alpha},$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_{10}]$. Assume that the data of the unknowns are given to us at the points $(x_j, t_k), j = 1, 2, \cdots, n, k = 1, 2, \cdots, m$. Now, we define feature vectors $f_1, f_2, f_3, \cdots, f_{10}$ as, $f_1 = [1, \ldots 1, \ldots, 1]^T$, $f_2 = [\mathcal{U}(x_1, t) \ldots \mathcal{U}(x_j, t) \ldots \mathcal{U}(x_n, t)]^T$, $\ldots$, $f_{10} = [\mathcal{U}_x(x_1, t)\mathcal{U}_{xx}(x_1, t), \ldots, \mathcal{U}_x(x_j, t)\mathcal{U}_{xx}(x_j, t), \ldots, \mathcal{U}_x(x_n, t)\mathcal{U}_{xx}(x_n, t)]^T$. Now the system of equations become,

$$V(t;\alpha) = P(t)\boldsymbol{\alpha}, \tag{5}$$

where V be the vector, which is the combination of time fractional derivative term and source term and P be the matrix of all features vector. Let $\boldsymbol{\beta} = (\alpha, \boldsymbol{\alpha})$ are the unknown parameter in the Equation (5). In [1], author ended with the system of Equations (5) with $\alpha = 1$, which was linear in unknown parameter. Here, we have a non-linear equation in unknown parameter $\boldsymbol{\beta}$, we can't solve the above equation with inverse or pseudo-inverse due to non-linearity. Also data may contains noise, and the problem can ill-posed. To tackle the ill-posedness, we use a regularizer with the entire data set. We use LASSO method, which is use L_1-regularization to promote the sparsity in the vector $\boldsymbol{\beta}$ is define as:

$$\min_{\boldsymbol{\beta}=(\alpha,\boldsymbol{\alpha})} \mathcal{J}(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{m} \|V(t_i,\alpha) - P(t_i)\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{6}$$

where $\lambda > 0$ is a regularization parameter. The issue with the Schaeffer's approach how to use the method in the case of a non-linear system of equations due to fractional order. Therefore, we use differential evolution to solve the Equations (6).

2.2 Differential Evolution

In 1996, the differential evolution method was introduced by Storn and Price [13, 11]. It is a stochastic, production-based optimization method for solving non-linear optimization problems. We used differential evolution as an optimization method for our problem. Let us consider the optimization problem (6). Here we aim to find the vector $\boldsymbol{\beta}^*$, which minimizes the $\mathcal{J}$. Suppose population size is N_p. Therefore, the population matrix can be written as

$$\boldsymbol{\beta}_n^g = [\beta_{n,1}^g, \beta_{n,2}^g, \cdots, \beta_{n,D}^g]. \tag{7}$$

Here D is the number of parameters, g is the generation and $n = 1, 2, \cdots, N_p$ and population matrix is generated with the help of prior information available of the parameters. If we do not have any prior information regarding parameter then we generate the population matrix with help of uniform random variable. For more details, see [11]. Now, we describes our algorithm for learning the parameter vector $\boldsymbol{\beta}$ presented in Equation (6).

Algorithm 1 Learning the parameter involve in the Equation (6)

```
Step 1 : With the help of given data, construct the feature matrix.
Step 2 : For the first generation, initialize the population matrix, and initialize all other parameters.
Step 3 : Execute the mutation and crossover steps for the given population.
Step 4 : With the help of LASSO operator evaluate error values for all the population members.
Step 5 : Execute the selection step with the help of error values.
Step 6 : Until the population members lie under a specific threshold value, repeat steps 2-5.
Step 7 : When population converges, return the final value.
```

3 Simulations and Numerical Experiments

In this Section, we illustrate the methodology which we have discussed in Section 2 with the help of the following three test Examples. The first two Examples show algorithm robustness at different level of noise. The third Example shows the advantage of the algorithm in the case of prior information regarding the parameters. The simulations were run on an Intel Core i5 $-$ 1135G7, 1.80GHzx4 machine with 16GB RAM.

Example 31 *Consider the time fractional diffusion equation*

$${}_C D_{0,t}^{\alpha} \mathcal{U}(x,t) = \mathcal{U}_{xx}(x,t) + \mathcal{G}(x,t), \tag{8}$$

$$\mathcal{G}(x,t) = \frac{24}{\Gamma(5-\alpha)} t^{4-\alpha} \sin \pi x, \ \mathcal{U}(x,0) = 0, \ \mathcal{U}(0,t) = 0, \ \textit{and}\ \mathcal{U}(1,t) = 0.$$

It can be checked that for $\alpha = 0.5$, $\mathcal{U}(x,t) = t^4 \sin \pi x$ be the exact solution of (8). Data is generated with the help of the exact solution of the equation with

Table 1: The identified terms and α for Example 31 at different level of noise.

	α	Identified Term	Coefficients
No Noise	0.4942	$\mathcal{U}_{xx}$	0.9996
2% Noise	0.4879	$\mathcal{U}_{xx}$	1.0000
5% Noise	0.4851	$\mathcal{U}_{xx}$	1.0000

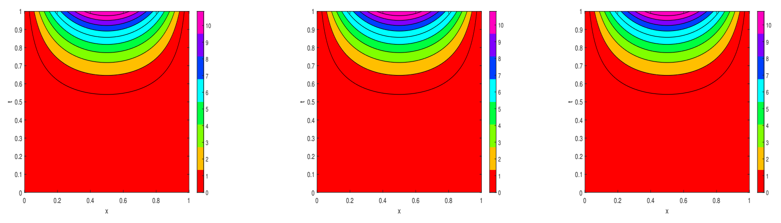
the 200×200 grid points and the feature matrix are obtained using third order Taylor's series expansion. Table 1 demonstrate the result of identifying the terms and fractional order in the underlying time fractional diffusion equation at different noise level. From the Table 1, one can observe that our proposed algorithm for identifying the time fractional diffusion equation is robust to noise. Figure 1 shows that the learned dynamic of $\mathcal{G}(x,t)$ is close to the original $\mathcal{G}(x,t)$ at a different noise level due to the learned fractional order.

Example 32 *Consider the time fractional Burgers equation*

$$ {}_{C}D^{\alpha}_{0,t}\mathcal{U}(x,t) = \mathcal{U}(x,t)\mathcal{U}_x(x,t) + 0.1\mathcal{U}_{xx}(x,t) + \mathcal{G}(x,t), \tag{9} $$

$$ \mathcal{G}(x,t) = \frac{10}{\Gamma(2-\alpha)} t^{1-\alpha} \sin \pi x - 100t^2 \pi \sin \pi x + 10t^2\pi^2 \sin \pi x, $$
$$ \mathcal{U}(x,0) = 0 = \mathcal{U}(0,t) = \mathcal{U}(1,t). $$

Table 2 demonstrate the result of identifying the terms and fractional order in the underlying time fractional Burgers equation at different noise level. Since the diffusion term's coefficient (viscosity) is small compared to the other term coefficients, from Table 1, one can observe that the viscosity term identifies as correctly as other terms involved in the time-fractional Burgers equation. Hence, our algorithm is efficient in identifying the terms having the coefficient with large value as well as terms having comparatively small value.



(a) Exact $\mathcal{G}(x,t)$

(b) Identified $\mathcal{G}(x,t)$ with 2% noise

(c) Identified $\mathcal{G}(x,t)$ with 5% noise

Fig. 1: Exact and Identified $\mathcal{G}(x,t)$ with different level of noise.

Example 33 *Consider the time fractional advection diffusion equation*

$$ {}_{C}D^{\alpha}_{0,t}\mathcal{U}(x,t) = \mathcal{U}_x(x,t) + \mathcal{U}_{xx}(x,t), \tag{10} $$

Table 2: The identified terms and α for Example 32 at different level of noise.

	α	Identified term	Coefficient
No Noise	0.4997	$\mathcal{U}\mathcal{U}_x,\mathcal{U}_{xx}$	0.9999,0.1001
2% Noise	0.4998	$\mathcal{U}\mathcal{U}_x,\mathcal{U}_{xx}$	0.9997,0.1001
5% Noise	0.4995	$\mathcal{U}\mathcal{U}_x,\mathcal{U}_{xx}$	0.9998,0.1003

$$\mathcal{U}(x,0) = e^{2x},\ \mathcal{U}(0,t) = e^{6t},\ \ and\,\mathcal{U}(1,t) = e^{2+6t}.$$

For simplicity we take $\alpha = 1$ and it can be checked that $\mathcal{U}(x,t) = e^{2x+6t}$ be the exact solution of (10). In this example, we will show the efficiency of our algorithm in the two cases: First, if we have prior information regarding coefficients of the terms in underlying advection-diffusion equations, whether it is a real number or integers. Second, if we do not have any information regarding coefficients. Assuming the

Table 3: The identified terms, CPU time and number of iteration required to convergence for Example 33 with two cases.

Have prior information				Do not have information			
Iteration	CPU time	Term	Coefficient	Iteration	CPU time	Term	Coefficient
14	13.52	$\mathcal{U}_x,\mathcal{U}_{xx}$	1,1	578	530.55	$\mathcal{U}_x,\mathcal{U}_{xx}$	1 ,1

coefficients of the term in the underlying advection-diffusion equation are integers (one can observe from the Equation (10)), while in the second case, we do not have any information regarding coefficients then the algorithm changes accordingly. Table 3 demonstrates the result of identifying the terms in the underlying time fractional advection diffusion equation, CPU time in second, and the number of iterations required to the convergence of the algorithm in the two cases. From the Table 3, one can observe that when we have some information regarding the coefficients of the term, our algorithm converges very fast compared to the second case. In the Schaeffer [1] approach we do not have this kind of advantage.

4 Conclusions

To summarize, this work introduces a computational framework based on LASSO method and differential evolution to discover the time fractional differential equations. This work is an extension of the recent study done by Schaeffer [1] to the case of time fractional differential equations. In [1], the author used the indirect optimization method to solve the optimization problem. The issue with [1] is how to use the method in the case of a non-linear system of equations due to fractional order. To address this issue in this paper, we have used the direct optimization method (differential evolution). An extension of this study could be generalizing this work for time-space fractional differential equations, which remains a major conceptual challenge due to fractional Taylor series approximation and space fractional order.

Acknowledgements

The First author acknowledges the support provided by University Grants Commission (UGC), India, under the grant number 20/12/2015(ii)EU-V. The second author acknowledges the support provided by the SERB India, under the grant number SERB/F/3060/2021 − 2022. The third author acknowledges the financial support from the North-Caucasus Center for Mathematical Research under agreement number 075 − 02 − 2021 − 1749 with the Ministry of Science and Higher Education of the Russian Federation.

References

1. H. Schaeffer, Learning partial differential equations via data discovery and sparse optimization, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 473 (2197) (2017) 20160446.
2. M. Mehra, R. K. Mallik, Solutions of differential–difference equations arising from mathematical models of granulocytopoiesis, Differential Equations and Dynamical Systems 22 (1) (2014) 33–49.
3. M. Raissi, P. Perdikaris, G. E. Karniadakis, Machine learning of linear differential equations using gaussian processes, Journal of Computational Physics 348 (2017) 683–693.
4. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational physics 378 (2019) 686–707.
5. A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Advances in Neural Information Processing Systems 34 (2021).
6. K. Srivastava, M. Ahlawat, J. Singh, V. Kumar, Learning partial differential equations from noisy data using neural networks, in: Journal of Physics: Conference Series, Vol. 1655, IOP Publishing, 2020, p. 012075.
7. A. Alikhanov, M. Beshtokov, M. Mehra, The crank-nicolson type compact difference schemes for a loaded time-fractional hallaire equation, Fractional Calculus and Applied Analysis 24 (4) (2021) 1231–1256.
8. M. Gulian, M. Raissi, P. Perdikaris, G. Karniadakis, Machine learning of space-fractional differential equations, SIAM Journal on Scientific Computing 41 (4) (2019) A2485–A2509.
9. G. Pang, L. Lu, G. E. Karniadakis, fpinns: Fractional physics-informed neural networks, SIAM Journal on Scientific Computing 41 (4) (2019) A2603–A2626.
10. A. K. Singh, M. Mehra, S. Gulyani, A modified variable-order fractional sir model to predict the spread of covid-19 in india, Mathematical Methods in the Applied Sciences (2021). doi:https://doi.org/10.1002/mma.7655.
11. A. K. Singh, M. Mehra, S. Gulyani, Learning parameters of a system of variable order fractional differential equations, Numerical Methods for Partial Differential Equations (2021). doi:https://doi.org/10.1002/num.22796.
12. S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the national academy of sciences 113 (15) (2016) 3932–3937.
13. R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_8