

# Data-Driven Discovery of Partial Differential Equations

Zihan Zhou - Warwick Mathematics Institute  
Supervisor: Dr Radu Cimpanu



## Introduction

Partial differential equations(PDE), commonly regarded as a valuable tool in describing physical and biological studies, are used to mathematically formulate how a system works[1]. Traditionally the governing PDEs are derived analytically, but this has its limitations when it comes to certain complex systems. Data-driven methods allow researchers to discover the governing PDEs based solely on the collected data, which are often measurements taken at different times or positions. This project aims to explore the data-driven method by testing them on three basic types of PDEs: the advection equation, the diffusion equation, and Burger's equation.

## Mathematical Models

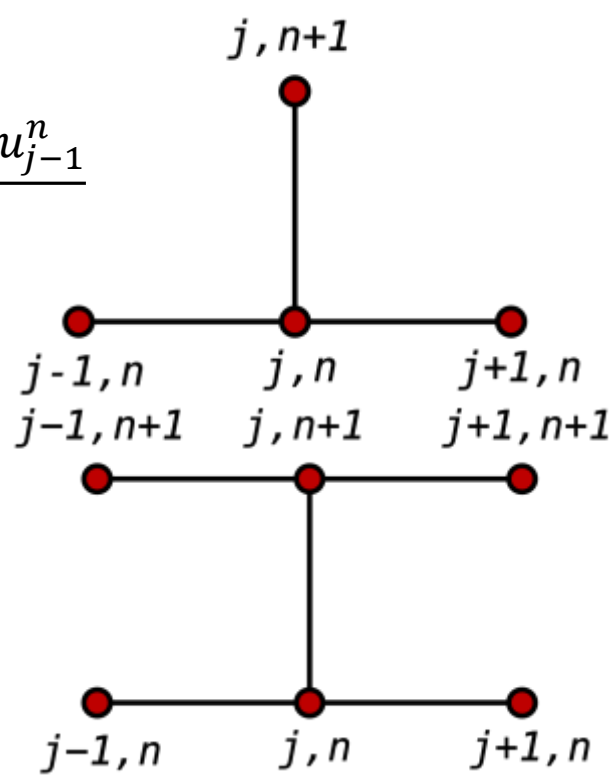
The study of each PDE is divided into three steps:  
Step 1. Derive numerical solution. Use finite difference methods(FDM) to obtain the numerical solution of the PDE, which gives the data needed for the data-driven method. This project uses two kinds of FDM: Forward difference for time derivative, central difference for space derivative(FTCS) and Crank-Nicholson method.

- FTCS:  $u_j^n = u(j\Delta x, n\Delta t)$

$$u_t \approx \frac{u_j^{n+1} - u_j^n}{\Delta t}, u_x \approx \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}, u_{xx} \approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{2\Delta x^2}$$

- Crank-Nicholson:

$$u_t \approx \frac{u_j^{n+1} - u_j^n}{\Delta t}$$
$$uu_x \approx \frac{1}{2} \left( u_j^n \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x} + u_j^{n+1} \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} \right)$$
$$u_{xx} \approx \frac{1}{2} \left( \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{2\Delta x} + u_j^{n+1} \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} \right)$$



Step 2. Apply data-driven method. Feed the data generated in step 1 to PDE-FIND[2], which will return a PDE that is supposed to govern the data.  
Step 3. Compare and calculate error. Compare the original PDE with the PDE given by the PDE-FIND program. Check if the program gives a PDE with the correct term, and if so, calculate the errors of the parameters.

## The Advection Equation

The advection equation is a simple representation of how a scalar quantity u (like temperature, concentration, etc.) is transported by a fluid moving with a constant velocity c in a single spatial dimension. x and t denotes position and time respectively. The equation is given by

$$u_t + cu_x = 0, \quad x \in \left[-\frac{L}{2}, \frac{L}{2}\right], t \in [0, T]$$

Set Gaussian initial condition and periodic boundary condition:

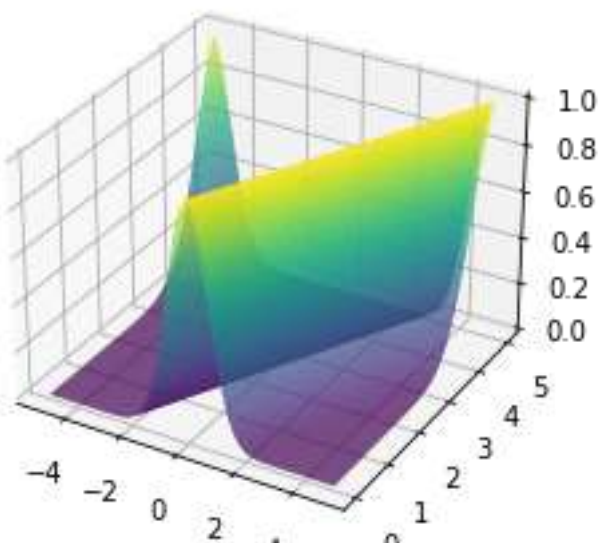
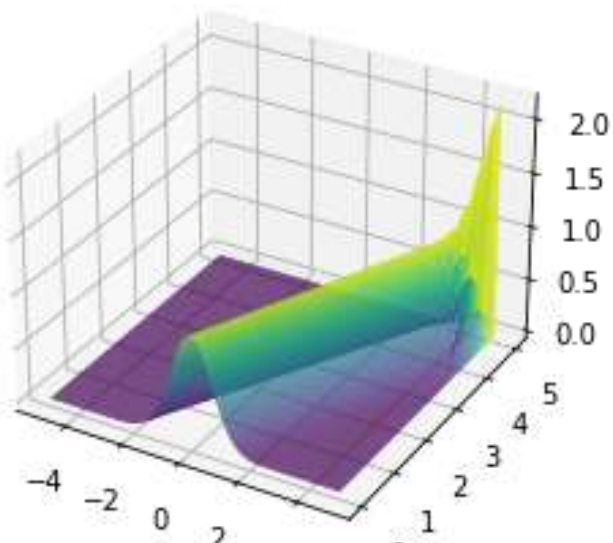
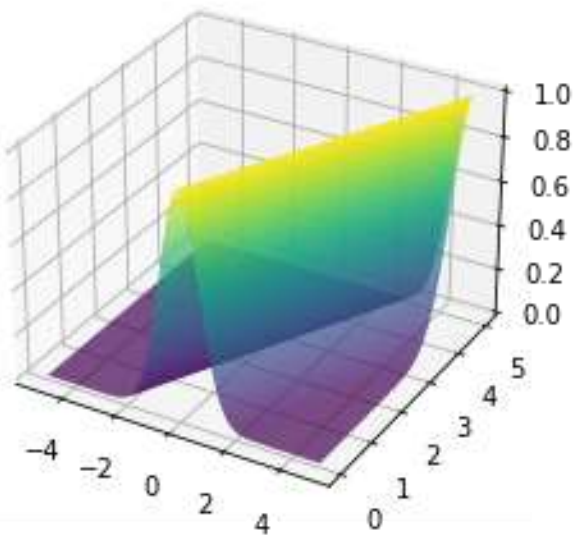
$$u(x, 0) = e^{-x^2}, \quad u\left(-\frac{L}{2}, t\right) = u\left(\frac{L}{2}, t\right)$$

Generate its exact solution, and use FTCS and Crank-Nicholson to obtain numerical solutions. Note that FTCS is unstable in this case and errors will accumulate, while Crank-Nicholson is stable. Set c = 1. Use the generated data as input for PDF-FIND and check and compare results.

Exact Solution

Explicit Solution

Crank-Nicholson Solution



PDE derived:

$$u_t = -1.004763u_x$$

$$u_t = 0.000030uu_{xxx}$$

$$u_t = -0.999448u_x$$

With the unstable FTCS scheme, PDE-FIND failed to produce the correct result. With the exact solution and the Crank-Nicholson the result is roughly the same as the original advection equation

## The Diffusion Equation

The diffusion equation, also known as the 1D heat equation, describes how heat (or similar diffusive scalar quantity) is distributed over time in a one-dimensional system. The heat equation deals with the spreading or dissipation of heat due to diffusion. The equation is given by

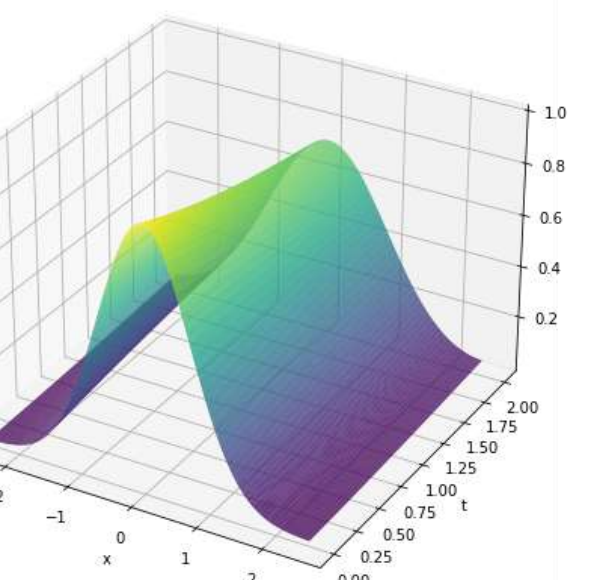
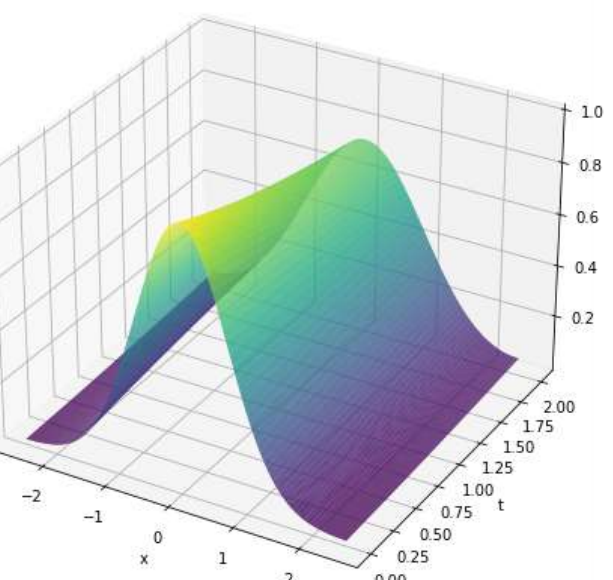
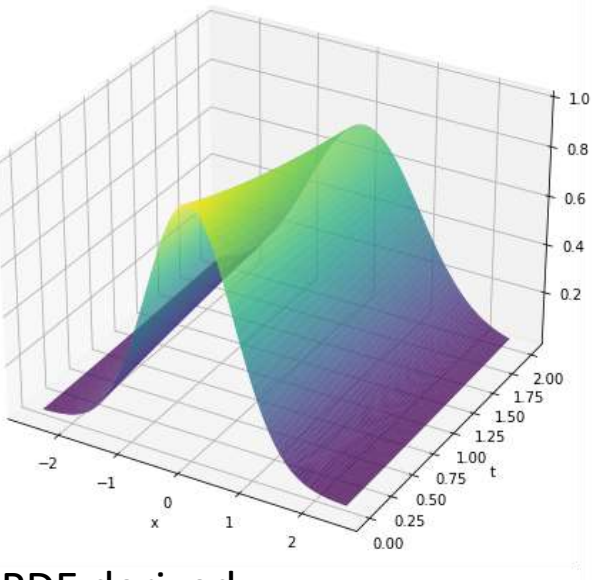
$$u_t = Du_{xx}, \quad x \in \left[-\frac{L}{2}, \frac{L}{2}\right], t \in [0, T]$$

where u is the concentration or temperature at a given position x and time t, and D is the diffusion coefficient. Set the same Gaussian initial condition and periodic boundary condition as before. Set D = 0.1. Here FTCS is stable under certain condition\* and Crank-Nicholson is unconditionally stable.

Exact Solution

Explicit Solution

Crank-Nicholson Solution



PDE derived:

$$u_t = 0.100084u_{xx}$$

$$u_t = 0.100149u_{xx}$$

$$u_t = 0.100003u_{xx}$$

## Burger's Equation

Burgers' equation is a fundamental partial differential equation from the field of fluid mechanics and nonlinear dynamics. It can be considered as a simplified model for studying shock waves, turbulence, and other phenomena in fluids. The equation is given by

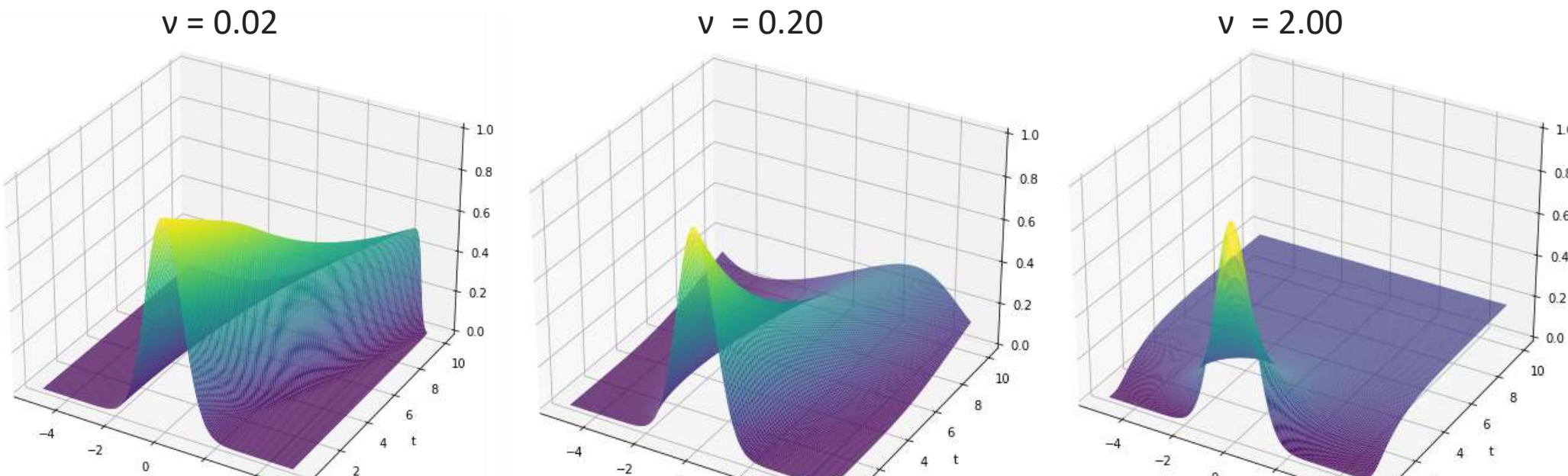
$$u_t + uu_x = \nu u_{xx}, \quad x \in \left[-\frac{L}{2}, \frac{L}{2}\right], t \in [0, T]$$

Where u can be thought of as velocity in a fluid flow context, t is time, x is the spatial coordinate, \nu is viscosity or diffusion coefficient. Set the same Gaussian initial condition and periodic boundary condition as before.

It is difficult to solve viscous Burger's equation analytically, and here Crank-Nicholson method is applied to obtain a numerical solution, which has been proved to be consistent and of order two in time and space[3].

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + \frac{1}{4\Delta x} [u_j^n (u_{j+1}^{n+1} - u_j^{n+1}) + u_j^{n+1} (u_j^{n+1} - u_{j-1}^{n+1})]$$
$$= \frac{\nu}{2\Delta x^2} (u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1} + u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

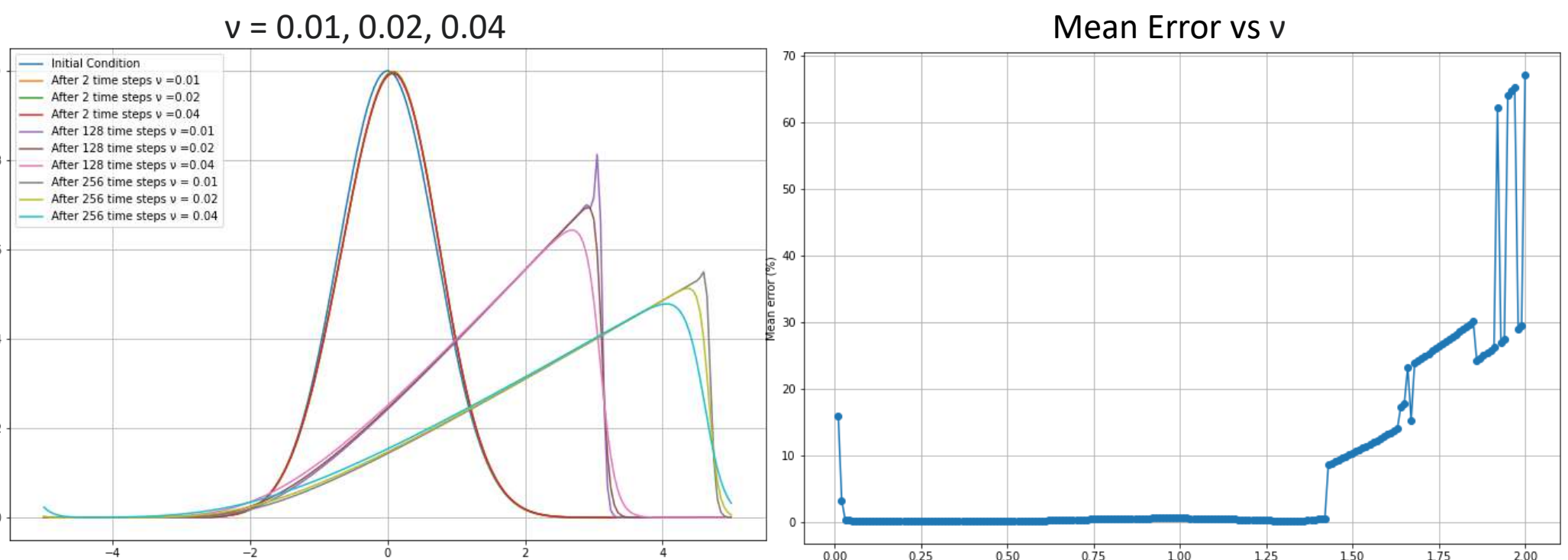
The shape of the surface depends largely on the value of \nu, and here we look into three different values: 0.02, 0.2, 2.0.



$$u_t = -1.057625uu_x + 0.283156u^2u_x \quad u_t = -1.000166uu_x + 0.199643u_{xx} \quad u_t = 0.000286 + 0.039955u - 0.296344u^2 - 0.457437u^3 + \dots$$

When \nu = 0.02, a shock is formed, and PDE-FIND fails to discover the correct terms. When \nu = 0.2, the surface is rather smooth and PDE-FIND returns roughly the right equation. When \nu = 2, the function diffuses too quickly for PDE-FIND to find the correct terms. It is natural to conjecture that PDE-FIND works better when the function is smooth with smaller diffusion term \nu. To investigate the condition for PDE-FIND to work, we test PDE-FIND on 200 different values of \nu = 0.01, 0.02, ..., 1.99, 2.00, and create a table containing \nu, the discovered PDEs, and the errors. Below are selected parts of the table.

\nu	PDE	Mean parameter error	Std. of parameter error
0.01	$u_t = (0.050140 + 0.000000i)u + (-0.163713 + 0.000000i)u^2 + (0.100025 + 0.000000i)u^3 + \dots$	16.009310%	14.174193%
0.02	$u_t = (-1.057625 + 0.000000i)uu_x + (0.283156 + 0.000000i)u^2u_x + \dots$	3.319493%	2.442960%
0.03	$u_t = (-0.995775 + 0.000000i)uu_x + (0.029581 + 0.000000i)u_{xx} + \dots$	0.420690%	0.001847%
0.04	$u_t = (-0.997450 + 0.000000i)uu_x + (0.039687 + 0.000000i)u_{xx} + \dots$	0.283914%	0.028941%
...	...	...	...
1.41	$u_t = (-0.995066 + 0.000000i)u_x + (1.409464 + 0.000000i)u_{xx} + \dots$	0.514596%	0.021155%
1.42	$u_t = (-0.994757 + 0.000000i)u_x + (1.419375 + 0.000000i)u_{xx} + \dots$	0.574683%	0.050394%
1.43	$u_t = (-0.998046 + 0.000000i)uu_x + (1.447058 + 0.000000i)u_{xx} + (-0.312211 + 0.000000i)uu_x + \dots$	8.626920%	8.431511%
1.44	$u_t = (-0.997805 + 0.000000i)uu_x + (1.457516 + 0.000000i)u_{xx} + (-0.320027 + 0.000000i)uu_x + \dots$	8.867872%	8.648358%
...	...	...	...
1.99	$u_t = (-0.034076 + 0.000000i)u + (0.260786 + 0.000000i)u^2 + (-0.391764 + 0.000000i)u^3 + \dots$	29.460429%	16.816766%
2.00	$u_t = (0.000286 + 0.000000i) + (-0.039955 + 0.000000i)u + (0.296344 + 0.000000i)u^2 + \dots$	67.171925%	20.811909%



The table shows that for \nu between 0.01 and 1.42, PDE-FIND manages to retrieve the correct terms, with a parameter error less than 2%. Graphs at the cutoff points shows between and \nu = 0.02 with \nu = 0.04, the function smooths out and PDE-FIND becomes accurate. When \nu goes above 1.43, PDE-FIND begins to pick up unrelated terms, and the mean parameter error sees a sharp rise. Now set \nu = 0.20, and try to identify the same dynamic with some added noise. Add a random noise with weighting \lambda = 10^{-6}, 10^{-5}, ..., 0.1, 1 to test the noise-sensitivity of PDE-FIND.

\lambda	PDE	Mean parameter error	Std. of parameter error
0.000001	$u_t = (-1.004325 + 0.000000i)uu_x + (0.199513 + 0.000000i)u_{xx} + \dots$	0.459872%	0.027403%
0.000010	$u_t = (-1.004322 + 0.000000i)uu_x + (0.199513 + 0.000000i)u_{xx} + \dots$	0.459751%	0.027505%
0.000100	$u_t = (-1.004292 + 0.000000i)uu_x + (0.199496 + 0.000000i)u_{xx} + \dots$	0.466512%	0.037324%
0.001000	$u_t = (-1.005010 + 0.000000i)uu_x + (0.180168 + 0.000000i)u_{xx} + \dots$	10.166434%	9.665440%
0.010000	$u_t = (-1.002290 + 0.000000i)uu_x + (0.584189 + 0.000000i)u_{xx} + \dots$	100.114487%	99.885513%
0.100000	$u_t = (0.006246 + 0.000000i) + (0.038874 + 0.000000i)u + (-0.184946 + 0.000000i)u^2 + \dots$	122.070662%	77.081559%
1.000000	$u_t = (-0.104636 + 0.000000i)u + (1.114702 + 0.000000i)u^2 + (-2.715334 + 0.000000i)u^3 + \dots$	150.000000%	50.000000%

PDE-FIND only gives the right result for noise values up to weighting of 10^{-4}, and fails for a bigger weighting index.

## Reference & Acknowledgement

- [1] Champion, Kathleen, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. 2019. "Data-Driven Discovery of Coordinates and Governing Equations." *Proceedings of the National Academy of Sciences* 116 (45): 22445–51. <https://doi.org/10.1073/pnas.1906995116>.
- [2] Rudy, Samuel H., Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. 2017. "Data-Driven Discovery of Partial Differential Equations." *Science Advances* 3 (4): e1602614. <https://doi.org/10.1126/sciadv.1602614>.
- [3] Wani, Sachin S., and Sarita H. Thakar. 2013. "Crank-Nicolson Type Method for Burgers Equation." *International Journal of Applied Physics and Mathematics*, 324–28. <https://doi.org/10.7763/ijapm.2013.v3.230>

\* When applied to the diffusion equation, FTCS is numerically stable if and only if  $\Delta t \leq \frac{\Delta x^2}{2D}$ .

This project is part of the 2023 URSS programme funded by University of Warwick. I would like to thank Dr Radu Cimpanu for his support in supervising this project.