AMSI VACATION RESEARCH SCHOLARSHIPS 2019–20

EXPLORE THE MATHEMATICAL SCIENCES THIS SUMMER

PDE Discovery with Machine Learning

Sean McGowan

Supervised by Dr Michael Chen

University of Adelaide

Vacation Research Scholarships are funded jointly by the Department of Education and Training and the Australian Mathematical Sciences Institute.

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

Australian Government Department of Education

Contents

1 Introduction **3**

2 Partial differential equations **4**

3 Method **4**

3.1 Sparse regression 5

3.2 Numerical differentiation 5

4 Simulating Data **6**

5 Results **7**

5.1 Validation 7

5.1.1 Heat equation 8

5.1.2 Burgers' equation 9

5.1.3 Kuramoto-Sivashinsky equation 10

5.2 Robustness 10

5.2.1 Preprocessing 11

5.2.2 Sensitivity analysis 11

5.3 Higher-order time derivatives 11

5.3.1 Wave equation 12

6 Alternate method **13**

7 Conclusion **14**

8 Acknowledgements **14**

9 References **15**

A MATLAB Code **16**

List of Figures

1 Validation framework 7
2 Heat equation with sine initial condition and constrained boundaries 8
3 Burgers' equation with flat initial condition, one constrained boundary and one oscillating boundary 9
4 KS equation with sine initial condition and constrained boundaries 10
5 Wave equation with flat initial condition and oscillating boundaries 12
6 Method of lines example 16

List of Tables

1 Heat equation comparison 8
2 Burgers' equation comparison 9
3 Kuramoto-Sivashinsky equation comparison 10
4 Wave equation comparison 12

Abstract

Partial differential equations (PDEs) are powerful tools for describing a variety of physical phenomena including electrodynamics, fluid dynamics and quantum mechanics. It is critical to develop techniques to model such systems in terms of PDEs so that underlying mechanisms of physical processes can be studied, and their behaviour predicted. Typically, models can be synthesised by first principles analysis of a system's dynamics, however most real life systems are too complex or intricate to derive accurate governing equations. It is important to develop methods to address this issue since fields such as control engineering, predictive modelling and simulation require correct equations to accurately formulate control algorithms or graphical solutions for these systems. Recently, neural networks and other machine learning techniques have been utilised to formulate PDE models for given data-sets. These methods have been enabled through the availability of inexpensive and fast computational simulations, and by ubiquity of large data-sets. This report will detail investigations into this procedure, focusing on the optimisation of this method and analysing its effectiveness.

1 Introduction

Accurate and interpretable models are necessary in describing a variety of physical systems within science, engineering and applied mathematics. Due to their versatility in describing such systems, partial differential equations (PDEs) are often used to formulate these models. Typically these PDEs are derived through use of complex mathematical tools such as first principles analysis, variational formulation or with conservation arguments, for example, the Navier-Stokes equations that govern fluid mechanics are derived through the conservation of mass and energy. The effectiveness of these tools relies on the ease and simplicity of the underlying mathematics, and thus these methods may not be suited for more complex systems. In the last five years, following advancements in computing and abundance of data, machine learning has been used to derive these PDE models from a given data-set. Throughout this report, we will explore using sparse regression to discover underlying PDEs and optimise these methods.

2 Partial differential equations

Partial differential equations are equations involving unknown multi-variable functions and their partial derivatives. These are powerful tools in modelling physical systems and have been used to describe electrodynamics, fluid mechanics and quantum mechanics. A general form of a PDE is

$$f(x_1, x_2, ..., x_n, u_{x_1}, u_{x_2}, ...u_{x_n}, ...) = 0.$$

These equations can also be written as *differential operators* to unknown functions. Differential operators are operators that accept functions as inputs and return functions through use of differentiation.

$$\mathcal{N}(u) = 0.$$

For example, consider the wave equation,

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u.$$

This equation can equivalently be written as,

$$\mathcal{N}(u) = 0, \text{ where } \mathcal{N} = \frac{\partial^2}{\partial t^2} - c^2 \nabla^2.$$

The process of solving a partial differential equation is called solving the PDE. An initial condition and a set of boundary constraints are required to determine the solution to a given PDE and this defines an initial boundary value problem.

3 Method

There are two main approaches to data driven discovery of PDEs; using neural networks and using sparse regression. Due to the Universal approximation theorem, neural networks can be universal approximators to continuous functions and have been used to accurately discover the differential operator of data-set's PDEs (Raissi, 2018). Despite the high accuracy and versatility of this method, the neural network acts as a black box in describing the PDE and therefore lacks the interprability of a closed-form model. In a common alternative approach (Rudy, 2016; Brunton, 2016; Berg, 2019), sparse regression is used to determine the coefficients of terms in a dictionary of possible terms to approximate the underlying differential operator. This results in a closed-form approximation of the differential operator which allows for subsequent analysis of the dynamics such as the conserved quantities or symmetries of the system. This closed-form differential operator also allows us to explore the dynamics of the system through changing initial and boundary conditions, and to then accurately predict future behaviour.

3.1 Sparse regression

Linear regression is an approach to solving the matrix equation $\mathbf{Ax} = \mathbf{b}$ by minimising a cost function $L = ||\mathbf{Ax} - \mathbf{b}||_2$ through iteration. We can alter this regression model by adding regularisation parameters to our cost function. If the L^0-norm of $\mathbf{x}$ is combined into the regression's cost, the resulting architecture is a sparse regression model,

$$L = ||\mathbf{Ax} - \mathbf{b}||_2 + \lambda||\mathbf{x}||_0.$$

In this approach, we seek to solve this matrix equation but also seek $\mathbf{x}$ to be a *sparse matrix*. A sparse matrix has relatively few non-zero elements. These objects are important in applications where data needs to be compressed to save storage, to increase speed and efficiency, or as in our case, to not over-fit our data. This ensures our resulting solution is parsimonious and involves the fewest assumptions. In the approach outlined by Rudy, given a data-set $\mathbf{u}$, the governing PDE is discovered through sparse regression (2016). The 'velocities' (first time derivatives of $\mathbf{u}$) are approximated and stored in the $\mathbf{b}$ matrix and the $\mathbf{A}$ matrix is populated by a dictionary of terms expected to appear in our differential operator. Sparse regression is then used to solve for ξ.

$$\mathbf{u}_t = \mathbf{\Theta}\xi.$$

The resulting non-zero entries of ξ are the coefficients of the corresponding dictionary terms, and with these a closed form solution of the discovered PDE can be written.

3.2 Numerical differentiation

In order to approximate the partial derivatives of a given data-set needed to populate these matrices, numerical differentiation is required. Finite difference or polynomial interpolation are robust enough for most problems however, as we may be calculating high-order derivatives, taking powers of derivatives and multiplying terms, any errors in these derivative approximations will compound through calculation and therefore throughout this project Chebyshev interpolation has been used. This approach to numerical differentiation is more well-conditioned than these other methods and utilises Chebyshev polynomials, defined by the following recurrence relation:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Chebyshev polynomials form a complete orthogonal system and are therefore a basis for the set of piece-wise smooth and continuous functions. These polynomials are fitted to the given data-set and then symbolically differentiated to give derivative approximations at the grid points. The fitting of these polynomials is the only computationally involved step as polynomial differentiation is notoriously simple.

4 Simulating Data

To validate this method's effectiveness at discovering PDEs we require synthetic data. To generate this data from equations we already know we need methods to solve given PDEs. While for some equations, analytic solutions exist and can be found using separation of variables, method of characteristics or change of variables, we generally require a numerical approach. Due to it's versatility, the method of lines was used throughout the project for the majority of PDEs. The technique of method of lines involves discretising all independent variables in the domain except time and treating the resulting problem as a system of coupled ODEs and integrating. The code for this approach is in the Appendices.

5 Results

5.1 Validation

The following diagram describes the validation framework used throughout this project. It involves solving a given PDE in MATLAB, sending this data to Python to approximate derivatives and calculate dictionary terms and then performing sparse regression in Python to yield a discovered PDE. The effectiveness of this approach can then simply be analysed through comparison of the initial and final PDEs' coefficients.

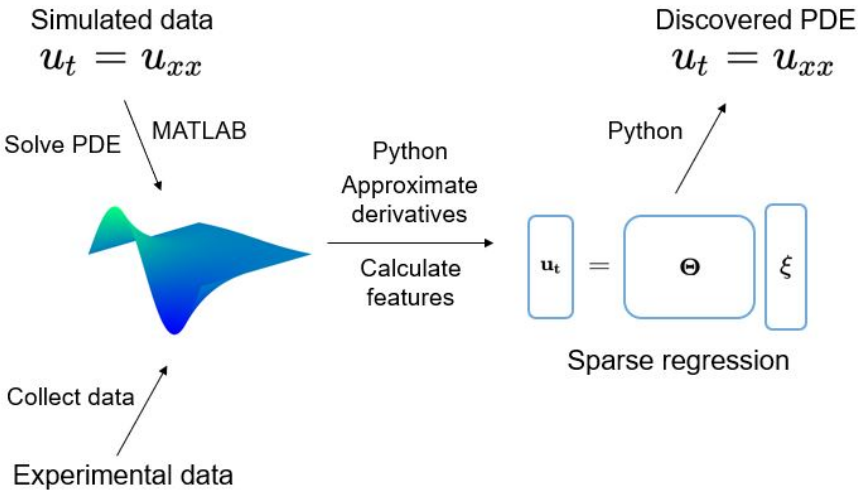

Figure 1: Validation framework

5.1.1 Heat equation

The heat equation was derived and solved by Fourier in 1822 to describe heat flow through solid body mediums. For $u(x,t)$, this equation is $u_t = u_{xx}$. This is one of the simplest first order time PDEs and was initially used to validate the sparse regression approach.

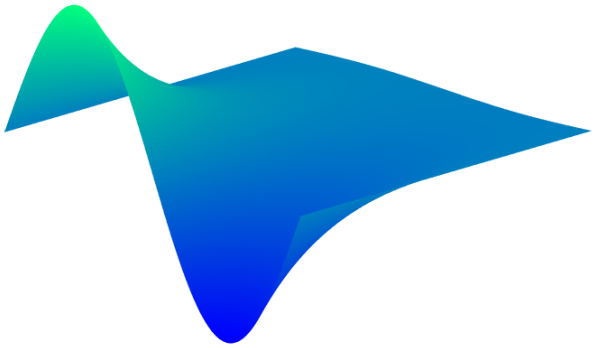

Figure 2: Heat equation with sine initial condition and constrained boundaries

Table 1: Heat equation comparison

Given PDE	Discovered PDE	Coefficient error
$u_t = u_{xx}$	$u_t = 1.003958u_{xx}$	0.4%

This sparse regression method produces the expected result and with minimal error for this simple case. This discovered PDE also behaves in a similar way when presented with the same initial and boundary conditions.

5.1.2 Burgers' equation

Burgers' equation was derived by Bateman in 1915 to model a variety of dynamical phenomena such as fluid mechanics, gas dynamics and acoustics. For $u(x,t)$, this equation is $u_t = -uu_x + \nu u_{xx}$. This PDE involves a nonlinear term, uu_x, and therefore this equation was used to validate the approach's effectiveness of discovering nonlinear terms.

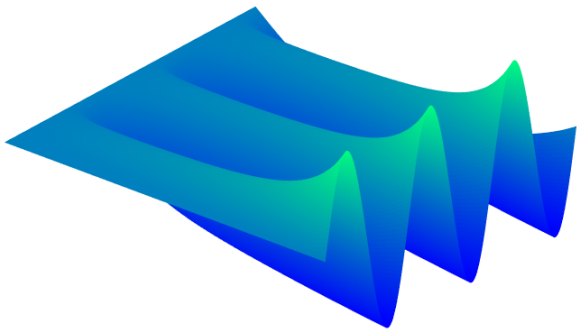

Figure 3: Burgers' equation with flat initial condition, one constrained boundary and one oscillating boundary

Table 2: Burgers' equation comparison

Given PDE	Discovered PDE	Coefficient error
$u_t = -uu_x + 0.1u_{xx}$	$u_t = -1.006738uu_x + 0.10111u_{xx}$	0.7%, 1.1%

The Burgers' equation was also discovered accurately to a high precision. Both the nonlinear term and the viscous term coefficients displayed similar errors of approximately 1%.

5.1.3 Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky equation was derived in 1977 to model the diffusive instabilities in laminar flame fronts. For $u(x,t)$, this equation is $u_t = -uu_x - u_{xx} - u_{xxxx}$. It features a fourth order spatial derivative and is known for it's notably chaotic behaviour. This equation was therefore used to assess the prediction of chaotic behaviour as well as the discovery of higher order derivative terms.

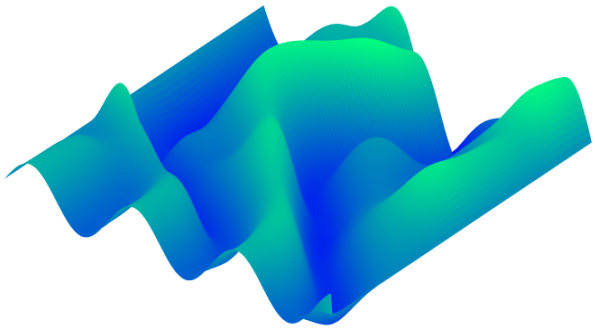

Figure 4: KS equation with sine initial condition and constrained boundaries

Table 3: Kuramoto-Sivashinsky equation comparison

Given PDE	Discovered PDE	Coefficient error
$u_t = -uu_x - u_{xx} - u_{xxxx}$	$u_t = -1.028904uu_x - 1.1333083u_{xx} - 1.163357u_{xxxx}$	2.9%, 13.3%, 16.3%

The KS equation was correctly discovered using sparse regression yet to a significantly lower precision. These errors are expected to arise from numerical inaccuracies involved in solving this PDE in MATLAB.

5.2 Robustness

Within the input data-sets it is expected that small variations to the underlying PDE solution will exist. This might be due to experimental disturbances within empirical data or numerical noise in PDE solving and ideally these variations should not disrupt the discovery process. Through preprocessing the data and undertaking sensitivity analysis to understand important structures within these data-sets we can optimise our methods to be robust to noise.

5.2.1 Preprocessing

To limit the effect of noise and increase the robustness of our method a preprocessing stage is required. In this step, the data is smoothed through calculating a moving average over each independent variable of the matrix elements. Although this process removes some of the data's information, it was found to be effective in resolving the noise enough to allow PDE discovery.

5.2.2 Sensitivity analysis

To better understand this method, sensitivity analysis was performed to determine the information within the data that was important for accurate discovery. This technique involves adding random noise to specific structures of simulated data and then assessing the regressions ability to discover the PDE to establish the structure's importance. This analysis determined that intricate and complex structures within data-sets are the most significant in discovery the underlying PDEs. These complex structures, such as wave reflections on boundary conditions, aid in removing the uncertainty of other similar PDEs also matching the data. There exists a push and pull between parsimony and disambiguation of the discovered PDE. As sparse regression will always seek the parsimonious and simplest solution we must include complex unique dynamics to disambiguate from simpler models.

For example, consider the viscous Burger's equation

$$u_t = -uu_x + \nu u_{xx}$$

If no viscous effects are present in the generated data, then in seeking parsimony, sparse regression will discover the inviscid Burger's equation

$$u_t = -uu_x$$

Therefore, in ensuring all dynamical behaviour is discovered through data-driven discovery, all complex dynamics are required to exist within the presented data-set.

5.3 Higher-order time derivatives

The Python code presented by Rudy has shown accuracy and versatility in discovering a range of PDEs, however it can currently only discover single time order PDEs (2016). We have extended this code so that higher-order time derivative terms can be discovered.

5.3.1 Wave equation

The standard second-order time PDE is the wave equation, studied by d'Alembert, Euler, Bernoulli and Lagrange, is an equation that describes the flow of waves. In general this PDE is of the form $u_{tt} = \nabla^2 u$ but in 1D it is $u_{tt} = u_{xx}$. This description is used to model water waves, sound waves and electromagnetic waves.

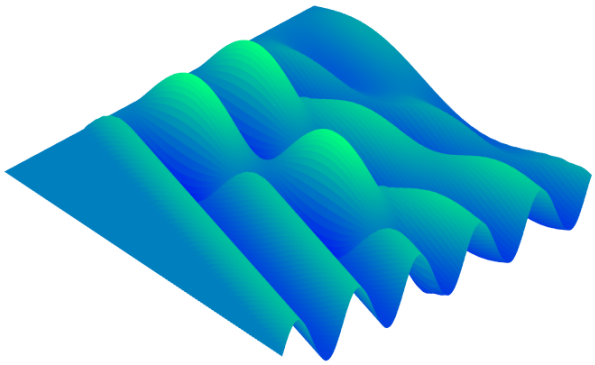

Figure 5: Wave equation with flat initial condition and oscillating boundaries

Table 4: Wave equation comparison

Given PDE	Discovered PDE	Coefficient error
$u_{tt} = u_{xx}$	$u_{tt} = 1.002377 u_{xx}$	0.2%

The extended code was able to accurately and precisely discover the wave equation in 1D, and in 2D to an equivalent precision. This addition to the current approach allows higher-order time derivative terms to be included and therefore a much wider range of PDEs to be discovered.

6 Alternate method

The sparse regression approach outlined previously has been shown to be effective in discovering PDEs of the form,

$$\frac{\partial^k u}{\partial t^k} = \mathcal{N}(u) \text{ for some k} \in \mathbb{N}$$

To discover PDEs of this form, we must assume the order k or trial a range of k to minimise error and maximise parsimony of our model. This is inefficient for data-driven discovery of PDEs as we should not require any prior assumptions or information in order to accurately discover the underlying dynamics. We propose an alternate method, where the regression model is

$$\mathbf{0} = \mathbf{\Theta}\xi$$

$$\text{With cost function, } L = ||\mathbf{\Theta}\xi||_2 + \lambda_1||\xi||_0 + \lambda_2(||\xi||_2 - 1)$$

In this approach, the $\mathbf{\Theta}$ matrix is populated by u and its partial derivatives (including mixed partials), as well as powers and combinations of these terms. This extends the dictionary of possible terms for the differential operator to include multiple time terms, higher order time derivatives and mixed partials. Therefore less assumptions are required for the model through these extensions.

The accuracy of the resulting solution depends on the accuracy of the derivative approximations and the terms inputted into $\mathbf{\Theta}$. If an infinite number of spatial derivatives, powers and combined terms are included in the dictionary, any differential operator can be determined. Consider the Sine-Gordon equation,

$$u_{tt} - u_{xx} + \sin(u) = 0$$

This nonlinear PDE is an alteration to the Klein-Gordon equation, where the u is replaced by $\sin(u)$. In order to discover the correct differential operator, enough terms must be included in the dictionary to uncover the Taylor series expansion of $\sin(u)$.

$$u_{tt} - u_{xx} + \sin(u) = u_{tt} - u_{xx} + u - \frac{u^3}{3!} + \frac{u^5}{5!} - ... = 0$$

In a similar way, any term involving functions of u can be discovered through including a sufficient number of powers of u to uncover the function's Taylor series approximation.

7 Conclusion

This project has investigated the technique of sparse regression to discover partial differential equations within data. This method has been shown to be efficient in discovering single time term PDEs including nonlinear terms. Sensitivity analysis demonstrated the importance of intricate, unique dynamics within the data to disambiguate from other PDEs to accurately discover the correct equation. An alternate method was presented that extends this previous technique and allows less assumptions in the resulting model. This approach is currently future work and further optimisation and implementation is required.

8 Acknowledgements

I thank Dr Michael Chen and Dr Simon Tuke for their assistance with this project and AMSI and the University of Adelaide for their support.

9 References

[1] Berg, J, Nystrom, K 2019, 'Data-driven discovery of PDEs in complex datasets', Journal of Computational Physics, vol. 384, pp. 239-252

[2] Brunton, S, Proctor, J, Kutz, N 2016, 'Discovering governing equations from data by sparse identification of nonlinear dynamical systems', PNAS, vol. 113

[3] Raissi, M 2018, 'Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations', Journal of Machine Learning Research, vol. 19, pp. 1-24

[4] Raissi, M, Karniadakis, G 2018, 'Hidden physics models: Machine learning of nonlinear partial differential equations', Journal of Computational Physics, vol. 357, pp. 125-141

[5] Rudy, S, Brunton, S, Proctor, J, Kutz, N 2016, 'Data-driven discovery of partial differential equations', Pattern Formation and Solitons

Appendices

A MATLAB Code

```
function kgwave
% Compute solutions of PDE
% discretised in space. Store u in odd and v
% in even numbered entries in uv vectors.
global j x
nPoints=200;
x=linspace(0,1,nPoints)';
j=2:nPoints-1;
uv0=nan(2*length(j),1);
% initial conditions
uv0(1:2:end)=0*x(j);
uv0(2:2:end)=0*x(j);
% integrate
[t,uv]=ode15s(@duvdt,linspace(0, 4, 200),uv0);
u=[uleft(t) uv(:,1:2:end) uright(t)];
% define RHS
function uvt=duvdt(t,uv)
global j x
dx=x(2)-x(1);
u=[uleft(t);uv(1:2:end);uright(t)];
uvt=nan(2*length(j),1);
uvt(1:2:end)=uv(2:2:end);
%finite difference approximation of RHS
uvt(2:2:end)=(u(j-1)-2*u(j)+u(j+1))/dx^2;
% define boundary conditions
function ua=uleft(t)
ua=zeros(size(t));
function ub=uright(t)
ub=zeros(size(t));
ub=sin(8.*t);
```

Figure 6: Method of lines example