Journal of Scientific Computing (2024) 98:13
https://doi.org/10.1007/s10915-023-02401-4

Feature Engineering with Regularity Structures

Ilya Chevyrev[1] · **Andris Gerasimovičs**[2] · **Hendrik Weber**[3]

Received: 18 January 2023 / Revised: 1 August 2023 / Accepted: 29 October 2023 /
Published online: 23 November 2023

Abstract
We investigate the use of models from the theory of regularity structures as features in machine learning tasks. A model is a polynomial function of a space–time signal designed to well-approximate solutions to partial differential equations (PDEs), even in low regularity regimes. Models can be seen as natural multi-dimensional generalisations of signatures of paths; our work therefore aims to extend the recent use of signatures in data science beyond the context of time-ordered data. We provide a flexible definition of a model feature vector associated to a space–time signal, along with two algorithms which illustrate ways in which these features can be combined with linear regression. We apply these algorithms in several numerical experiments designed to learn solutions to PDEs with a given forcing and boundary data. Our experiments include semi-linear parabolic and wave equations with forcing, and Burgers' equation with no forcing. We find an advantage in favour of our algorithms when compared to several alternative methods. Additionally, in the experiment with Burgers' equation, we find non-trivial predictive power when noise is added to the observations.

Keywords Regularity structures · Path signatures · Regression · Supervised learning · Partial differential equations

Contents

✉ Ilya Chevyrev
ichevyrev@gmail.com

Andris Gerasimovičs
andrisger@gmail.com

Hendrik Weber
hendrik.weber@uni-muenster.de

[1] The University of Edinburgh, Edinburgh, UK

[2] University of Bath, Bath, UK

[3] University of Münster, Münster, Germany

1 Introduction

The aim of this paper is to explore the effectiveness of models from Hairer's theory of regularity structures [16] as feature sets of space–time signals. A model is a collection of polynomial functions of the signal which has been used to great success in the analysis of stochastic partial differential equations (SPDEs). This paper is the first, to our knowledge, to explore its effectiveness in a machine learning context.

One of the motivations for this study comes from the fact that models are a higher-dimensional analogue of the *path signature*, a central object in Lyons' theory of rough paths [32]. The signature is the collection of the iterated integrals of a path, which has a rich mathematical structure; it is known to characterise the path up to a natural equivalence relation [2, 6, 19] and leads to a natural notion of non-commutative moments on pathspace [8, 11]. Over the past decade, the ability of the signature to encode information about a path in an efficient and robust way has made it a powerful tool in the analysis of *time-ordered data*. Examples of applications of signatures include the recognition of handwriting [15, 45] and gestures [33], analysis of financial data [24, 30], statistical inference of SDEs [37], analysis of psychiatric and physiological data [1, 35], topological data analysis [10], neural networks [23], and kernel learning [11, 25].[1] See [7] for a gentle introduction to the path signature and some of its early applications.

1.1 Our Contribution

Our main contribution is to introduce a novel concept of *model feature vector* (MFV) that provides an extension of path signatures outside the context of time-ordered data, that is, to data parameterised by *multi-dimensional* space. The MFV is a collection of space–time functions from D to $\mathbb{R}$, where $D \subset \mathbb{R}^d$ for $d \geq 0$, that is built from an input signal $\xi : D \to \mathbb{R}^K$. In the context of solving PDEs, the signal may incorporate a forcing term and boundary data. The motivation for the MFV is the fact that solutions to PDEs with a given forcing term and boundary data should be well-approximated at a space–time point z by components of the corresponding MFV evaluated at the same space–time point; we give further details in

[1] Graham [15], Morrill et al. [35] notably received first prizes in the ICDAR 2013 competition and the PhysioNet 2019 Computing in Cardiology Challenge respectively.

Sect. 2.1. In a machine learning context, the MFV provides a set of features of the original data that can be used in learning algorithms.

In addition to proposing the MFV, we give evidence that these features can carry important information in several test cases. The basic problem on which we test the use of MFV is:

Problem 1 *For a point $z \in \mathbb{R}^d$, predict the value $u(z)$, where u solves a PDE with a known forcing ξ and boundary condition u_0 but with unknown coefficients.*

We focus on the case that the PDE in question is an evolution equation. To address Problem 1, we propose two algorithms, Algorithms 1 and 2, in Sect. 2.3 based on elementary linear regression with MFVs in supervised learning tasks. Algorithm 1 is designed to predict $u(z)$ in the presence of a general forcing ξ, while Algorithm 2 is designed to work when there is no forcing (or equivalently $\xi = 0$) in which case one can leverage the flow property of u to improve predictability. An important feature of Algorithm 2 is that it predicts $u(t, x)$ for all space–time points (t, x), thereby effectively learning the entire function u.

We investigate the effectiveness of Algorithms 1 and 2 in numerical experiments in Sect. 3. We apply Algorithm 1 to non-linear parabolic and wave equations with forcing and fixed initial conditions, and apply Algorithm 2 to Burgers' equation with no forcing but varying initial condition.

In the case of Burgers' equation, Algorithm 2 performs similarly to an adaptation of the PDE-FIND algorithm [38] on noiseless data and data with small noise, and outperforms the latter on data with large noise (see Sect. 3.3.1). In the case of a parabolic equation, Algorithm 1 outperforms some basic off-the-shelf regression algorithms (SVR, K-Nearest Neighbours, Random forests) applied simply by treating the forcing as a large vector.

We emphasise that the definition of MFV in Sect. 2.2 and the algorithms in Sect. 2 are presented independently of PDEs and could be applied to learn other functions of the underlying signal, not necessarily the solution of a PDE—see Sect. 4 for further discussion.

1.1.1 Related Works

The MFV is inspired by the notion of a model from [16]. The main difference between our definition and that in [16] is that we suitably include the boundary data of the signal as part of the model. The path signature is a special case of MFV (Proposition 2.4).

The idea to apply machine and statistical learning methods to find, predict, or study solutions of PDEs has seen much attention in recent years. See for example [5, 18, 36, 38, 40, 44] and the references therein. We also mention the works [21, 22, 27, 29, 47] that, like ours, treat boundary data in machine learning-based solvers. Many works in this direction have focused on new design of learning algorithms. In contrast, our main contribution comes from designing a new set of features which can be used in a range of algorithms. As such, we expect our approach to complement many existing methods.

Since the appearance of this article, several works have built on MFVs (or related approaches) especially in combination with neural networks. See for example [20, 43].

2 Model Feature Vectors and Regression Algorithms

In this section we motivate and define the "model feature vector" and introduce two algorithms based on models for learning functions of space–time signals.

We denote by $\mathbb{N} = \{0, 1, 2, \dots\}$ the set of non-negative integers and by $\mathbb{R}$ the set of real numbers. Assume that we are given a spatial domain $D \subset \mathbb{R}^d$ for $d \geq 0$ and a time horizon

$T > 0$. Given a multi-index $a \in \mathbb{N}^d$, we denote $\partial^a = \partial_1^{a_1} \dots \partial_d^{a_d}$ where

$$\partial_i^{a_i} := \frac{\partial^{a_i}}{\partial x_i^{a_i}}, \quad \text{for } i = 1, \dots, d.$$

We will also define the order of a multi-index as $|a| := \sum_{i=1}^d a_i$. Note that $\partial^0 u = u$. We let ∂_t denote the partial derivative with respect the time parameter $t \in [0, T]$.

2.1 Motivation

Our motivating problem is to learn the solution of a PDE on $[0, T] \times D$ given by

$$\begin{aligned} \mathcal{L}u &= \mu(\{\partial^a u\}_{|a|\le q}) + \sigma(\{\partial^a u\}_{|a|\le q})\xi, \\ u(0, x) &= u_0(x), \end{aligned} \tag{2.1}$$

where $\mathcal{L}$ is a linear differential operator, $u_0 \colon D \to \mathbb{R}$ is the initial condition, and $\xi \colon [0, T] \times D \to \mathbb{R}$ is a forcing.[2] The functions $\mu, \sigma : \mathbb{R}^{1+d+\dots+d^q} \to \mathbb{R}$ take as arguments the partial derivatives of u up to order q (i.e. the jet of u to level q) and are assumed to be smooth and unknown, while (ξ, u_0) is known.

Such an equation is often called a *PDE with forcing* ξ. When ξ is a random function, e.g. space–time white noise, it is also referred to as a *stochastic* PDE (SPDE).

For this discussion, we assume $\mathcal{L} = \partial_t - \nu\Delta$ is the heat operator with viscosity $\nu > 0$, where $\Delta = \sum_{i=1}^d \partial_i^2$ is the Laplacian on $D \subset \mathbb{R}^d$, and μ, σ depend only on u (and not its derivatives). The case when μ, σ depend on $\partial^a u$ with $|a| > 0$ as well as the case of other choices of $\mathcal{L}$, e.g. the wave operator, are left to the reader.

Under good enough assumptions on the functions μ, σ, u_0 and the forcing ξ, equation (2.1) admits a local in time mild solution.[3] In particular, Picard's Theorem implies that u is the limit of the following recursive sequence

$$\begin{aligned} u^{(0)} &= I_c[u_0], \\ u^{(n+1)} &= I_c[u_0] + I[\mu(u^{(n)})] + I[\sigma(u^{(n)})\xi], \end{aligned} \tag{2.2}$$

where the operators I and I_c are defined by

$$\begin{cases} (\partial_t - \nu\Delta) I[f] = f, \\ I[f](0, \cdot) = 0, \end{cases} \quad \begin{cases} (\partial_t - \nu\Delta) I_c[g] = 0, \\ I_c[g](0, \cdot) = g, \end{cases} \tag{2.3}$$

for functions $f \colon [0, T] \times D \to \mathbb{R}$ and $g \colon D \to \mathbb{R}$, subject to the same boundary conditions as in (2.1).

The idea is now to Taylor expand the function μ up to m terms and the function σ up to ℓ terms in the equation for $u^{(n+1)}$. Define $u^{0,m,\ell} = I_c[u_0]$ and recursively set

$$u^{n+1,m,\ell} = I_c[u_0] + \sum_{k=0}^{m} \frac{\mu^{(k)}(0)}{k!} I[(u^{n,m,\ell})^k] + \sum_{k=0}^{\ell} \frac{\sigma^{(k)}(0)}{k!} I[(u^{n,m,\ell})^k \xi]. \tag{2.4}$$

Then, heuristically, since Taylor's expansion implies $u^{n,m,\ell} \to u^{(n)}$ as $m, \ell \to \infty$ and since Picard theorem implies $u^{(n)} \to u$ as $n \to \infty$, we see that $u^{n,m,\ell}$ should be a good candidate for approximating u.

[2] One might need to include other initial information like an initial speed for the case of the wave equation from Sect. 3.2.

[3] In the case that ξ is white noise on $[0, T] \times [0, 1]$, smoothness of the above functions is enough (see [12]).

It is not difficult to see from (2.4) that $u^{n,m,\ell}$ is a polynomial function of $I_c[u_0]$ and ξ that involves iterated integrals (i.e. iterated applications of I). Recalling that the unknowns are μ and σ, and thus $\mu^{(k)}$ and $\sigma^{(k)}$ are also unknown, it is sensible to encode as features these polynomials of $I_c[u_0]$ and ξ and learn the solution map $(u_0, \xi) \mapsto u$ via linear regression. Our definition of "model feature vector" below precisely encodes this collection of polynomials that appear in $u^{n,m,\ell}$ in a more general setting. These polynomials closely resemble models appearing in the theory of regularity structures, see [16, Sec. 8], which is the motivation behind our terminology.

2.2 Model Feature Vectors

We now generalise and abstract the polynomial features discussed in the previous subsection. Fix for the rest of this section a pair $(\{u^{(i)}\}_{i\in\mathcal{J}}, \xi)$ (an "observed signal") where $\mathcal{J}$ is a finite index set (possibly empty) and

$$\xi = (\xi^{(1)}, \dots, \xi^{(K)}) \colon [0,T] \times D \to \mathbb{R}^K, \quad u^{(i)} \colon [0,T] \times D \to \mathbb{R}. \tag{2.5}$$

We call ξ the *forcing*. The case $d = 0$ corresponds to just $\xi \colon [0,T] \to \mathbb{R}^K$ and $u^{(i)} \colon [0,T] \to \mathbb{R}$. In the experiments in Sect. 3, we sometimes let u be fixed, so that the signal is only ξ, and sometimes we fix ξ (essentially taking $\xi \equiv 0$) so that the signal is only $u^{(i)}$. One should think of $\{u^{(i)}\}_{i\in\mathcal{J}}$ as "boundary conditions", like $I_c[u_0]$ in (2.2), and where we allow multiple boundary conditions (as needed, e.g. for the wave equation).

Let us fix a linear operator I that maps space–time functions $f : [0,T] \times D \to \mathbb{R}$ to space–time functions $I[f]$. For example, $I[f]$ could be a convolution with some space–time kernel or a solution to some linear PDE with forcing f.

Definition 2.1 Consider a tuple of non-negative integers $\alpha = (m, \ell, q) \in \mathbb{N}^3$ and $n \in \mathbb{N}$. The *model feature set* $\mathcal{S}_\alpha^n$ is the finite set of formal symbols defined inductively by[4] $\mathcal{S}_\alpha^0 = \mathcal{J}$ and

$$\mathcal{S}_\alpha^n = \Big\{\mathcal{I}\big[\Xi^{(k)} \prod_{i=1}^{p} D^{a_i}\tau_i\big] : 1 \le p+1 \le \ell\Big\} \cup \Big\{\mathcal{I}\big[\prod_{i=1}^{p} D^{a_i}\tau_i\big] : 1 \le p \le m\Big\} \cup \mathcal{S}_\alpha^{n-1},$$

where $a_i \in \mathbb{N}^d$ with $|a_i| \le q$, $1 \le k \le K$, and $\tau_i \in \mathcal{S}_\alpha^{n-1}$. Here $\Xi^{(k)}$ and D^{a_i} are formal symbols. When $a_i = 0$ we simply write $D^{a_i}\tau = \tau$.

The *model feature vector* (MFV, or simply *model*) $\mathcal{M}_\alpha^n$ of $(u^{(i)}, \xi)$ as in (2.5) is the family functions $\mathcal{M}_\alpha^n \colon \mathcal{S}_\alpha^n \to \mathbb{R}^{[0,T]\times D}$ that we denote by

$$\mathcal{M}_\alpha^n = (f_\tau)_{\tau\in\mathcal{S}_\alpha^n},$$

where $f_\tau \colon [0,T] \times D \to \mathbb{R}$ is defined recursively by $f_i = u^{(i)}$ for $i \in \mathcal{J}$ and for $\tau = \mathcal{I}\big[\Xi^{(k)} \prod_{i=1}^{p} D^{a_i}\tau_i\big]$ and $\sigma = \mathcal{I}\big[\prod_{i=1}^{p} D^{a_i}\sigma_i\big]$

$$f_\tau = I\big[\xi^{(k)} \prod_{i=1}^{k} \partial^{a_i} f_{\tau_i}\big], \qquad f_\sigma = I\big[\prod_{i=1}^{k} \partial^{a_i} f_{\sigma_i}\big]. \tag{2.6}$$

We call n the *height* of a model, m the *additive width*, ℓ the *multiplicative width*, and q the *differentiation order*. Furthermore,

[4] We use the convention $\prod_{i=1}^{0} \tau_i = 1$. Furthermore, the product of symbols is commutative, e.g. we identify $\mathcal{I}[\Xi D^1\tau_1 D^2\tau_2] = \mathcal{I}[D^2\tau_2 \Xi D^1\tau_1]$, and multiplication by 1 is the identity, i.e. $1\tau = \tau$.

- if $q = 0$, we call $\mathcal{M}^n_\alpha$ a *model without derivatives*,
- if $\ell = 0$, we call $\mathcal{M}^n_\alpha$ a *model without forcing*, and
- if $\mathcal{J}$ is empty, we call $\mathcal{M}^n_\alpha$ a *model without initial conditions*.

We will often use an abuse of notation and write $f_\tau \in \mathcal{M}^n_\alpha$ meaning that there exists a symbol $\tau \in \mathcal{S}^n_\alpha$ such that $\mathcal{M}^n_\alpha[\tau] = f_\tau$. The symbols in $\mathcal{S}^n_\alpha$ can be represented as decorated combinatorial rooted trees as in [3, Sec. 2].

Note that additive width m limits how many functions could be multiplied if none of them includes a component of the forcing ξ, while multiplicative width ℓ limits how many functions could be multiplied if one of them is a component of ξ.

Example 2.2 Consider $n = 1$, and $\alpha = (2, 2, 1)$, and $d = 1$. Suppose $\mathcal{J} = \{c\}$ is a singleton. Then, denoting $D_x = D^{(1)}$,

$$\mathcal{S}^1_\alpha = \big\{c, \mathcal{I}[\Xi], \mathcal{I}[c], \mathcal{I}[(c)^2], \mathcal{I}[\Xi c], \mathcal{I}[D_x c], \mathcal{I}[c D_x c], \mathcal{I}[(D_x c)^2], \mathcal{I}[\Xi D_x c]\big\}.$$

If we instead take $\bar\alpha = (2, 2, 0)$, i.e., consider the model without derivatives, then

$$\mathcal{S}^1_{\bar\alpha} = \big\{c, \mathcal{I}[\xi], \mathcal{I}[c], \mathcal{I}[(c)^2], \mathcal{I}[\xi c]\big\}.$$

Suppose now that $D = [0, 1]$. Let $\xi, u^{(c)} \colon [0, T] \times D \to \mathbb{R}$ be given by $\xi(t, x) = \sin(t)$ and $u^{(c)}(t, x) = \cos(x)$. Finally, suppose $\mathcal{I}[f](t, x) = \int_0^x f(t, y)\,\mathrm{d}y$ is the integration-in-space operator. Then the MFV $\mathcal{M}^1_{\bar\alpha}$ is

$$\mathcal{M}^1_{\bar\alpha} = \big\{\cos(x), x\sin(t), \sin(x), (x + \sin(x)\cos(x))/2, \sin(t)\sin(x)\big\},$$

where we used $\int_0^x \cos(y)\,\mathrm{d}y = \sin(x)$ and $\int_0^x \cos(y)^2\,\mathrm{d}y = (x + \sin(x)\cos(x))/2$. The space–time functions f_τ in $\mathcal{M}^1_{\bar\alpha}$ correspond to the symbols of $\mathcal{S}^1_{\bar\alpha}$ in the same order, e.g., $f_{\mathcal{I}[(c)^2]} = (x + \sin(x)\cos(x))/2$.

To give an example at level $n = 2$, one of the symbols in S^2_α is $\tau = \mathcal{I}[\Xi\mathcal{I}[c D_x c]]$. The corresponding function $f_\tau \in \mathcal{M}^2_\alpha$ is

$$\begin{aligned} f_\tau(t, x) &= \int_0^x \sin(t)\Big(\int_0^y \cos(z)(-\sin(z))\,\mathrm{d}z\Big)\,\mathrm{d}y = \int_0^x \sin(t)\frac{1}{2}(-\sin^2(y))\,\mathrm{d}y \\ &= \frac{1}{8}\sin(t)(\sin(2x) - 2x). \end{aligned}$$

We next show precisely how the path signature is generalised by the MFV. Consider $n \geq 1$ and a differentiable path

$$X = (X^{(1)}, \dots, X^{(K)}) \colon [0, T] \to \mathbb{R}^K.$$

Definition 2.3 The level-n *signature* of X over an interval $[s, t] \subset [0, T]$ is the collection of K^n numbers $\{S^I_{s,t}(X)\}_I$ indexed by multi-indexes $I = (i_1, \dots, i_n) \in \{1, \dots, K\}^n$ and defined by the iterated integrals

$$S^{(i_1,\dots,i_n)}_{s,t}(X) = \int_s^t \int_s^{t_n} \cdots \int_s^{t_2} \dot X^{(i_1)}_{t_1} \cdots \dot X^{(i_{n-1})}_{t_{n-1}} \dot X^{(i_n)}_{t_n}\,\mathrm{d}t_1 \dots \mathrm{d}t_{n-1}\,\mathrm{d}t_n. \tag{2.7}$$

Proposition 2.4 *Define the family of symbols $\mathcal{W}^n$ inductively by $\mathcal{W}^0 = \{1\}$ and*

$$\mathcal{W}^n = \{\mathcal{I}[\Xi^{(k)}\tau] : \tau \in \mathcal{W}^{n-1},\ 1 \leq k \leq K\},$$

where $\Xi^{(k)}1 := \Xi^{(k)}$. Then there is a bijection $\varphi \colon \{1, \dots, K\}^n \to \mathcal{W}^n$ given by

$$\varphi(i_1, \dots, i_n) = \mathcal{I}[\Xi^{(i_n)}\mathcal{I}[\Xi^{(i_{n-1})}\mathcal{I}[\dots\mathcal{I}[\Xi^{(i_1)}]\dots]]].$$

Consider furthermore $d = 0$ and define the $\mathbb{R}^K$-valued forcing

$$\xi = \{\xi^{(i)}\}_{i=1}^K \colon [0, T] \to \mathbb{R}^K \qquad \xi^{(i)} = \dot{X}^{(i)}.$$

Let $I[\xi](t) = \int_0^t \xi(s)\,\mathrm{d}s$ be the integration-in-time operator and $\alpha = (0, 2, 0)$ and $\mathcal{J} = \emptyset$ (i.e. consider the model without initial conditions).

Then $\mathcal{W}^n \subset \mathcal{S}_\alpha^n$ and, for all $(i_1, \dots, i_n) \in \{1, \dots, K\}^n$,

$$\mathcal{M}_\alpha^n[\varphi(i_1, \dots, i_n)](T) = S_{0,T}^{(i_1,\dots,i_n)}(X). \tag{2.8}$$

Proof The inclusion $\mathcal{W}^n \subset \mathcal{S}_\alpha^n$ and that $\varphi \colon \{1, \dots, K\}^n \to \mathcal{W}^n$ is a bijection are clear. Equality (2.8) is clearly true for $n = 1$ and in general follows by induction:

$$\begin{aligned}
\mathcal{M}_\alpha^n[\varphi(i_1, \dots, i_n)](T) &= \mathcal{M}_\alpha^n[\mathcal{I}[\Xi^{(i_n)}\varphi(i_1, \dots, i_{n-1})]](T) \\
&= \int_0^T \dot{X}^{(i_n)} S_{0,t_n}^{(i_1,\dots i_{n-1})}(X)\,\mathrm{d}t_n \\
&= S_{0,T}^{(i_1,\dots i_n)}(X)
\end{aligned}$$

where the second equality follows from the inductive hypothesis of (2.8) for $n - 1$ and the third equality follows readily from the definition (2.7). □

In our experiments below, the finite index set $\mathcal{J}$ and "boundary conditions" $\{u^{(i)}\}_{i\in\mathcal{J}}$ will be taken as follows: in Sect. 3.3 $\mathcal{J}$ will be a singleton $\mathcal{J} = \{c\}$ and $u^{(c)}$ will be the solution to the linear heat equation with a given initial condition $u_0^{(c)} \colon D \to \mathbb{R}$, i.e. $u^{(c)} = I_c[u_0^{(c)}]$ for I_c as in Sect. 2.1; in Sect. 3.1 $\mathcal{J}$ will primarily be empty $\mathcal{J} = \emptyset$ as we will ignore initial conditions (though see Sect. 3.1.2 for an exception); in Sect. 3.2 $\mathcal{J}$ will contain two elements $\mathcal{J} = \{c, s\}$ and $u^{(c)}$ (resp. $u^{(s)}$) will be the solution of the linear wave equation with initial condition $u_0^{(c)}$ and initial speed 0 (resp. initial condition 0 and initial speed $u_0^{(s)}$), where both $u_0^{(c)}, u_0^{(s)}$ are given.

While we consider only a space–time setting, Definition 2.1 readily adapts to an purely spatial setting. In this case the linear operator I would map functions $f \in \mathbb{R}^D$ to $I[f] \in \mathbb{R}^D$, e.g. $I[f](x) = \int_D K(x, y) f(y)\,\mathrm{d}y$ for a kernel $K \colon D \times D \to \mathbb{R}$.

2.3 Regression Algorithms

In this subsection, we propose two supervised learning algorithms which use the MFV of an input $(\{u^{(i)}\}_{i\in\mathcal{J}}, \xi)$ to learn an output u. While in principle there is no limitation of the nature of u (vector, classification label, etc.), we will consider the special case where u is a number associated to a space–time point or is a space–time function. In the experiments in Sect. 3, u will be the solution to a PDE with forcing ξ and a given initial condition. We will furthermore consider henceforth $K = 1$, so the forcing is $\mathbb{R}$-valued and simply write $\xi^{(1)} = \xi$; the generalisation $K > 1$ is left to the reader.

2.3.1 Prediction at One Point

In the following algorithm, one should think of the observation u as a quantity which depends on the signal $(\{u^{(i)}\}_{i\in\mathcal{J}}, \xi)$ at a given space–time point $(t, x) \in [0, T] \times D$. Below $\{u^{(i)}\}_{i\in\mathcal{J}}$ and ξ will denote functions $u^{(i)}, \xi \colon [0, T] \times D \to \mathbb{R}$ for every $i \in \mathcal{J}$.

Algorithm 1 ***(Prediction at one point.)***

Parameters: *integers* $n, m, \ell, q \in \mathbb{N}$ *and an operator* I.

Input:

- *a point* $(t, x) \in [0, T] \times D$;
- *a set* $\mathcal{J}$;
- *set of observed triplets* $(u, \{u^{(i)}\}_{i\in\mathcal{J}}, \xi) \in U^{\text{obs}}$ *where* $u \in \mathbb{R}$;
- *a set of pairs* $(\{v^{(i)}\}_{i\in\mathcal{J}}, \zeta) \in U^{\text{pr}}$ *for which we want to make a prediction.*

Output: *Prediction* $u^{\text{pr}} \in \mathbb{R}$ *for every* $(\{v^{(i)}\}_{i\in\mathcal{J}}, \zeta) \in U^{\text{pr}}$.

Step 1 *Let* $\alpha = (m, \ell, q)$. *For each* $(u, \{u^{(i)}\}_{i\in\mathcal{J}}, \xi) \in U^{\text{obs}}$ *and (resp. each* $(\{v^{(i)}\}_{i\in\mathcal{J}}, \zeta) \in U^{\text{pr}}$*) construct a model* $\mathcal{M}_\alpha^n = (f_\tau)_{\tau\in\mathcal{S}_\alpha^n}$ *using* $\{u^{(i)}\}_{i\in\mathcal{J}}$ *and* ξ *(resp.* $\{v^{(i)}\}_{i\in\mathcal{J}}$ *and* ζ*)*

Step 2 *Fit a linear regression of* u *against* $(f_\tau(t, x))_{\tau\in\mathcal{S}_\alpha^n}$ *for each* $(u, \{u^{(i)}\}_{i\in\mathcal{J}}, \xi) \in U^{\text{obs}}$.

Step 3 *For each* $(\{v^{(i)}\}_{i\in\mathcal{J}}, \zeta) \in U^{\text{pr}}$, *construct a prediction* u^{pr} *using the linear fit constructed from Step 2 and the associated model* $\mathcal{M}_\alpha^n$.

Recall that our motivating problem is to learn the solution of a PDE (2.1) at a given point (t, x) where (u_0, ξ) are observed but μ, σ are unknown. Using the notation of Sect. 2.1, our typical choice for $\mathcal{J}$ is a singleton $\mathcal{J} = \{c\}$ with $u^{(c)} = I_c[u_0]$ (but we use other choices if we wish to encode more or less boundary conditions). The heuristic reason why Algorithm 1 should work for predicting PDEs comes from the fact that functions in $\mathcal{M}_\alpha^n$ constructed from ξ and $I_c[u_0]$ well approximate the n-th Picard iterate $u^{(n)}$ which itself should converge to the solution of (2.1) for smooth μ and σ.

Remark 2.5 If it is known that the Eq. (2.1) is additive, i.e. that σ is a constant, then the heuristic of Sect. 2.1 suggests that one should consider $\mathcal{M}_\alpha^n$ with $\ell = 1$. More generally, if it is known that both μ and σ are polynomials, then the heuristic suggests that taking m and $\ell - 1$ greater than the respective degrees of μ and σ would likely not improve the accuracy of the above algorithm. These remarks follow from the fact that polynomials agree with their Taylor expansion (for high enough order of expansion).

Note that, in Algorithm 1, we regress against the functions in the model at one input space–time point (t, x) only (see Sect. 2.1 for the motivation behind this choice in the case of PDEs). In the case of path signatures (Definition 2.3), this corresponds to using only the endpoint T of the signature, i.e. $S^I_{0,T}(X)$, which is common practice (see e.g. [1, 10, 25]). There are situations, however, where it is beneficial to use the signature of a path over different segments, i.e. use $S^I_{s,t}$ as a feature for different choices of $[s, t] \subset [0, T]$, and the choice of segments is a hyperparameter, see e.g. the sliding window approach of [45]. It would be of interest to explore if a similar approach yields any benefit for MFVs.

2.3.2 Prediction Using Flow Property

We will now focus on predicting functions u defined on all space–time points which have a given initial condition and no forcing. Algorithm 2 below is designed to work when u satisfies the *time-homogeneous flow property*: $u(t, x)$ should depend on $u(0, \cdot)$ in the same way as $u(t + h, x)$ depends on $u(h, \cdot)$.

The algorithm employs a discretisation of time $\mathcal{O}_T = \{0 = t_0 < t_1 < \ldots < t_N = T\} \subset [0, T]$ which we assume is equally spaced, i.e. $t_k = \delta k$ where $\delta = T/N$. The observed and predicted functions of this algorithm are both functions $\mathcal{O}_T \times D \to \mathbb{R}$.

Assume further that we are given an additional linear map I_c which is an *initialising map*: given $u_0 \colon D \to \mathbb{R}$, $I_c[u_0]$ is another function $[0, \delta] \times D \to \mathbb{R}$. Let $\mathcal{M}_\alpha^n(u_0)$ be the model without forcing ($\ell = 0$) constructed on $[0, \delta] \times D$ with $\mathcal{J} = \{c\}$ and $u^{(c)} := I_c[u_0]$.

We briefly describe the algorithm in words. Suppose that we know or have a prediction for $u(t_k, x)$ for some $k \in \{0, \dots, N-1\}$ and all $x \in D$. Under the time-homogeneous flow property, it is natural to seek an approximation for $u(t_{k+1}, x)$ using a functional linear regression of the form

$$u(t_{k+1}, \cdot) \approx a(\cdot) + \sum_{\tau \in \mathcal{S}_\alpha^n} b_\tau(\cdot) f_\tau(\delta, \cdot), \tag{2.9}$$

where $a, b \colon D \to \mathbb{R}$ are functions to be learned and $f_\tau \in \mathcal{M}_\alpha^n(u_{t_k})$. The time homogenous flow property implies that a and b_τ are expected to only depend on the time step δ and not t_k (but a, b_τ can depend on $x \in D$). In the training phase, we therefore decompose each observation $u \colon \mathcal{O}_T \times D \to \mathbb{R}$ into N 'subobservations' $u(t_k, \cdot) \colon D \to \mathbb{R}$, for $k = 0, \dots, N-1$, and learn the coefficients a, b_τ from these subobservations. The prediction phase then recursively applies the formula (2.9) to predict u from the 'initial condition' $u(0, \cdot)$. In the following, we will sometimes write u_t for the function $u(t, \cdot) \colon D \to \mathbb{R}$.

Algorithm 2 ***(Prediction using flow property.)***

Parameters: *integers $n, m, q \in \mathbb{N}$, operator I, and initialising map I_c.*

Input:

- *a collection $\{u(t,x)\}_{(t,x) \in \mathcal{O}_T \times D} \in U^{\text{obs}}$ of observed functions;*
- *a collection $u_0 \in U^{\text{pr}}$ of initial conditions $u_0 \colon D \to \mathbb{R}$ for which we want to make a prediction.*

Output: *A prediction $u^{\text{pr}} \colon \mathcal{O}_T \times D \to \mathbb{R}$ for every $u_0 \in U^{\text{pr}}$.*

Step 1 *Let $\alpha = (m, 0, q)$. For $k = 0, \dots N-1$ and each $u \in U^{\text{obs}}$ construct a model $\mathcal{M}_\alpha^n(u_{t_k})$ on $[0, \delta] \times D$ with $\mathcal{J} = \{c\}$ and $u^{(c)} = I_c[u_{t_k}]$.*

Step 2 *For each $x \in D$ fit a linear regression as in (2.9) of*

$$(u(t_{j+1}, x))_{u \in U^{\text{obs}}, j = 0, \dots, N-1}$$

against

$$\big((f_\tau(\delta, x))_{f_\tau \in \mathcal{M}_\alpha^n(u_{t_j})}\big)_{u \in U^{\text{obs}}, j = 0, \dots, N-1} .$$

Step 3 *For each $u_0 \in U^{\text{pr}}$ construct a model $\mathcal{M}_\alpha^n(u_0)$ on $[0, \delta] \times D$ with $\mathcal{J} = \{c\}$ and $u^{(c)} = I_c[u_0]$. Make a prediction of $u^{\text{pr}}(t_1, x)$ for each $x \in D$ based on the fit from Step 2 and $(f_\tau(\delta, x))_{f_\tau \in \mathcal{M}_\alpha^n(u_0)}$.*

Step 4 *Recursive step. For each $u_0 \in U^{\text{pr}}$, $k \geq 1$, and the predicted $u_{t_k}^{\text{pr}}$, construct a model $\mathcal{M}_\alpha^n(u_{t_k}^{\text{pr}})$ on $[0, \delta] \times D$ with $\mathcal{J} = \{c\}$ and $u^{(c)} = I_c[u_{t_k}^{\text{pr}}]$ and make a prediction of $u^{\text{pr}}(t_{k+1}, x)$ for each $x \in D$ based on a linear fit from Step 2 and $(f_\tau(\delta, x))_{f_\tau \in \mathcal{M}_\alpha^n(u_{t_k}^{\text{pr}})}$.*

When specific boundary values are given, one might need to enforce these for the predicted function u^{pr}. For example one might set $u^{\text{pr}}(t_k, x) = 0$ for $x \in \partial D$ and every $k = 0, \dots N$ if this was known.

As remarked earlier, Algorithm 2 effectively converts the size of the training set for the linear fit from $|U^{\text{obs}}|$ to $N \times |U^{\text{obs}}|$.

Algorithm 2 aims to address a problem similar to that of learning a dynamical system. A different approach to this problem is dynamic mode decomposition, which is based on spectral analysis of the Koopman operator [39, 41].

2.3.3 Feature Selection and Hyperparameters

The cardinality of $\mathcal{S}_\alpha^n$ grows exponentially with n. To avoid overfitting or to speed up the learning, it can be important to restrict further the number of elements in $\mathcal{S}_\alpha^n$. We do this below by introducing a function called *degree* $\deg\colon \mathcal{S}_\alpha^n \to \mathbb{R}$ which satisfies $\deg(\xi) = \eta$ and $\deg(i) = \eta_i$, $i \in \mathcal{J}$, for some $\eta, \eta_i \in \mathbb{R}$ together with the inductive definition

$$\deg I[\tau] = \beta + \deg\tau, \quad \deg\partial^{a_i}\tau = \deg\tau - |a_i|, \quad \deg\prod_{i=1}^{k}\partial^{a_i}\tau_i = \prod_{i=1}^{k}\deg\partial^{a_i}\tau_i, \tag{2.10}$$

for some $\beta > 0$. This definition is set so that deg of symbols from $\mathcal{S}_\alpha^n$ will be usually larger for bigger n, m, ℓ, q. We will then perform the regression in our algorithms against the functions in the model whose symbol does not exceed a certain degree γ. When used, the degree function and cutoff γ are additional parameters in Algorithms 1 and 2.

We follow a "rule of thumb" of keeping the ratio (number of train cases):(number of predictors) above 10 (see [17, Sec. 4.4] and references therein for a discussion about such rules). Thus, one would choose γ so that the number of functions in $f_\tau \in \mathcal{M}_\alpha^n$ with $\deg\tau \le \gamma$ is at least 10 times smaller than number of elements in U^{obs}.

Note that it is much easier to keep the train cases to predictors ratio above 10 for Algorithm 2 because we have $N|U^{\text{obs}}|$ train cases compared to only $|U^{\text{obs}}|$ train cases in Algorithm 1. Nevertheless, it is still beneficial to use degree for Algorithm 2 for computational reasons. Indeed, the linear regression time complexity for Algorithm 2 is $O(|\mathcal{M}_\alpha^n|^3 + |\mathcal{M}_\alpha^n|N|U^{\text{obs}}|)$. Thus, a large size of the model can drastically slow down the learning.

The use of deg and cutoff γ is motivated by analysis of SPDEs,[5] but other choices of feature selection are possible and may lead to improved learning. For example, it is possible to consider higher degree features but with a sparsity (i.e. l_0 norm) penalty, which is often employed in dictionary learning, though this choice would still require the computation of a large model $\mathcal{M}_\alpha^n$, at least on the training data; it would be of significant interest to investigate this form of feature selection (we are not aware of any systematic studies of sparse dictionary learning even for signature features).

In addition to the degree, Algorithms 1 and 2 come with several further hyperparameters, one of which is the 'height' (number of iterated applications of a linear operator) of the model; in the case of path signatures (Definition 2.3), our 'height' is the 'level' of a signature. In all the numerical experiments in Sect. 3, it was established that using a model with a larger height improves the performance of regression. Another hyperparameter is the linear operator I used in the definition of a model. In the case of Burgers' equation analysed in Sect. 3.3, we additionally found that I can be 'guessed' from the data, yielding sensible results (the guess for I does not need to be precise but the precision influences the prediction power). Some further discussion is given in Sect. 4.

3 Numerical Simulations

We present several numerical experiments where we learn the solution of the PDE (2.1) with different choices of operator $\mathcal{L}$ and non-linearities μ, σ. In general one needs to specify the boundary conditions of (2.1), i.e. the values of $u(t, x)$ for $x \in \partial D$. For simplicity,

[5] The notion of degree is similar to the one introduced in [16] and is related to the Hölder regularity of functions in the model which are built from highly oscillatory signals.

we only consider periodic boundary conditions in our experiment, but Dirichlet or Neumann boundary conditions can be easily implemented. As in Sect. 2.3, we will only consider MFVs (Definition 2.1) with $K = 1$ and $q \leq 1$.

To approximate the continuum, we fix a finite grid $\mathcal{O} \subset [0, T] \times D$. We will assume that $\mathcal{O} = \mathcal{O}_T \times \mathcal{O}_X$ where $\mathcal{O}_T = \{0 = t_0 < t_1 < \cdots < t_N = T\}$ for some integer $N \geq 1$ and $\mathcal{O}_X$ is a finite grid of points in D. We will work with functions defined on the grid $\mathcal{O}$ instead of $[0, T] \times D$. For this purpose, the operator I, the partial derivatives ∂_i, and I_c (whenever it is used) must have approximations on $\mathcal{O}$.

In all experiments below we use an ordinary least squares linear regression. See

https://github.com/andrisger/Feature-Engineering-with-Regularity-Structures.git

for Python code containing implementation of the model and experiments from this section.

3.1 Parabolic PDEs with Forcing

In this subsection we will suppose that the differential operator in (2.1) is given by $\mathcal{L} = \partial_t - \nu\Delta$, where $\nu > 0$ is the viscosity and $\Delta = \sum_{i=1}^{d} \partial_i^2$ is the Laplacian on $D \subset \mathbb{R}^d$. This motivates the following definition.

Definition 3.1 Fix an initial condition $u_0 \colon D \to \mathbb{R}$ and a forcing $\xi \colon [0, T] \times D \to \mathbb{R}$ as well as $n, m, \ell \geq 1$ and $\nu > 0$. Let $q \leq 1$ and $\alpha = (m, \ell, q)$. The model $\mathcal{M}_\alpha^n$ for the parabolic equation with viscosity ν is constructed by taking $\mathcal{J} = \{c\}$ with $u^{(c)} = I_c[u_0]$ where operators I and I_c are given by (2.3).

Remark 3.2 Algorithm 1 does not require knowledge of μ or σ in (2.1). However, in the experiments in this subsection, μ and σ will be polynomials, and we will use knowledge of their degree to choose the hyperparameters m, ℓ. Another hyperparameter is ν since this determines I through (2.3). When μ, σ, ν are completely unknown, these hyperparameters could be chosen, as usual, by splitting the data into training, validation and test sets, and tuning the hyperparameters on the validation set. See also Sect. 3.3 where a starting point for an approximation of the viscosity $\tilde{\nu}$ is derived from the training data.

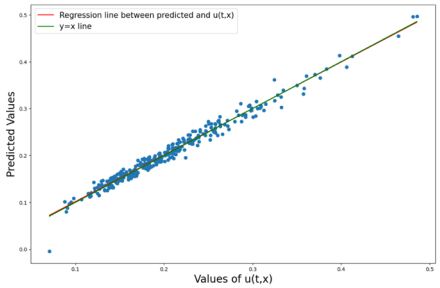
3.1.1 Multiplicative Forcing

Consider the following PDE

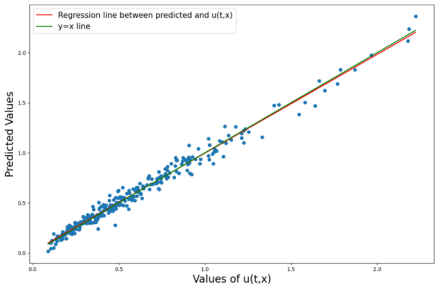
$$\begin{aligned} &(\partial_t - \Delta)u = 3u - u^3 + u\,\xi \quad \text{for}(t, x) \in [0, 1] \times [0, 1], \\ &u(t, 0) = u(t, 1) \quad \text{(Periodic BC)}, \\ &u(0, x) = x(1 - x), \end{aligned} \tag{3.1}$$

where ξ is a space–time forcing. Here we discretise space and time respectively in 100 and 1000 evenly distanced points, which we use to define the grids $\mathcal{O}_X$ and $\mathcal{O}_T$. We solve (3.1) for each forcing ξ using a finite difference method on the same discretisation $\mathcal{O} = \mathcal{O}_T \times \mathcal{O}_X$ (see [31, Sec. 10.5]).

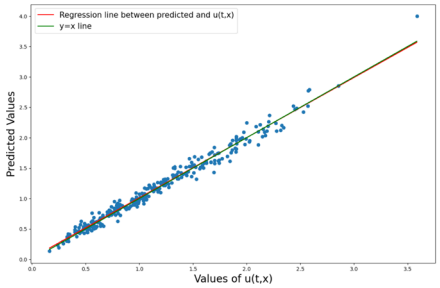
We take here ξ as approximations of space–time white noise. We performed Algorithm 1 both using the full model $\mathcal{M}_\alpha^n$ from Definition 3.1 with viscosity $\nu = 1$ and the model without the initial conditions (i.e. where $\mathcal{J}$ is assumed to be empty in the construction of $\mathcal{S}_\alpha^n$). We have found that in practice using the full model did not drastically improve the errors (see Remark 3.3). Therefore, we primarily present results in this subsection for the model without the initial condition.



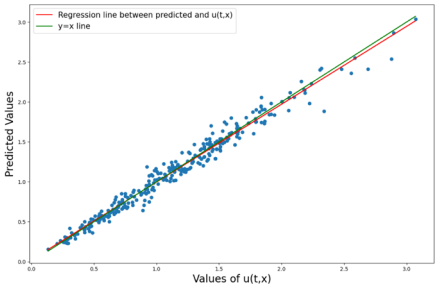
(a) Prediction at $(t, x) = (0.05, 0.5)$. Relative ℓ^2 error: 4.8%. Slope: 0.99. Error standard deviation: 4.1%. $R^2 = 0.98$.



(b) Prediction at $(t, x) = (0.5, 0.5)$. Relative ℓ^2 error: 7.8%. Slope: 0.99. Error standard deviation: 5.2%. $R^2 = 0.98$.



(c) Prediction at $(t, x) = (1, 0.5)$. Relative ℓ^2 error: 6.5%. Slope: 0.99. Error standard deviation: 5.5%. $R^2 = 0.98$.



(d) Prediction at $(t, x) = (1, 0.95)$. Relative ℓ^2 error: 7.0%. Slope: 0.97. Error standard deviation: 6.0%. $R^2 = 0.97$.

Fig. 1 Results of linear regression of solutions to (3.1) against the functions in model $\mathcal{M}_\alpha^4$ with $\alpha = (3, 2, 0)$, without initial conditions and of degree ≤ 5. The x-axis contains values of $u(t, x)$ for realisations of the forcing ξ from the test set and the y-axis contains predictions of the linear regression. Subplots **a**–**d** show predictions at space–time points $(t, x) = (0.05, 0.5)$, $(0.5, 0.5)$, $(1, 0.5)$, $(1, 0.95)$ respectively

We construct a model with $\mathcal{J} = \emptyset$ of height $n = 4$, additive width $m = 3$, multiplicative width $\ell = 2$,[6] and differentiation order $q = 0$ (because μ and σ do not depend on $\partial_i u$) so that $\alpha = (3, 2, 0)$. We assign a degree from Sect. 2.3.3 to satisfy (2.10) with $\beta = 2$ and $\deg\xi = -1.5$.[7] In the experiments below we only consider functions $f_\tau \in \mathcal{M}_\alpha^4$ with $\deg\tau \leq 5$.

We randomly sample 1000 realisations of approximations of white noise ξ on $\mathcal{O}$ and solve (3.1) for each realisation. We then split the pairs (u, ξ) into training and test sets of size 700 and 300 respectively. There are only 56 functions in $f_\tau \in \mathcal{M}_\alpha^4$ with degree $\deg\tau \leq 5$ thus corresponding to a ratio of training cases to the number of predictors of $700/56 = 12.5$. In Fig. 1, we show results of performing Algorithm 1 with models without initial conditions at various space–time points (t, x). In every subfigure one can see a scatter plot of actual values of $u(t, x)$ from the test set plotted against the predicted values. The error is measured

[6] See Remark 2.5 for a motivation behind taking these particular widths.

[7] This is motivated by the Hölder regularity of space–time white noise being $-1.5 - \varepsilon$ for any small $\varepsilon > 0$ and the fact that the heat operator I increases the Hölder regularity by 2.

Table 1 Average relative ℓ^2 errors, slopes, and R^2 for linear regression against models of different heights

Model height	Error (%)	Slope	R^2	Error (%)	Slope	R^2
	$(t, x) = (0.05, 0.5)$			$(t, x) = (0.5, 0.5)$		
1	9.3	0.91	0.91	21.1	0.85	0.84
2	5.4	0.97	0.97	9.5	0.97	0.97
3	4.9	0.98	0.97	8.0	0.98	0.98
4	4.8	0.98	0.98	7.7	0.98	0.98
	$(t, x) = (1, 0.5)$			$(t, x) = (1, 0.95)$		
1	23.2	0.73	0.73	22.1	0.75	0.75
2	13.7	0.91	0.91	13.4	0.91	0.91
3	7.7	0.97	0.97	7.7	0.97	0.97
4	6.5	0.98	0.98	6.6	0.98	0.98

Prediction is performed at space–time points $(t, x) = (0.05, 0.5), (0.5, 0.5), (1, 0.5), (1, 0.95)$

as a relative ℓ^2 error, i.e. for the vector of realisations R and predictions P we set

$$\|R - P\|_{\bar{\ell}^2} := \sqrt{\frac{1}{n}\sum_{i=1}^{n} |R_i - P_i|^2} \quad \text{and} \quad \|R\|_{\bar{\ell}^2} := \sqrt{\frac{1}{n}\sum_{i=1}^{n} |R_i|^2}.$$

and the relative ℓ^2 error is defined by

$$E := Error(R, P) = \frac{\|R - P\|_{\bar{\ell}^2}}{\|R\|_{\bar{\ell}^2}} = \sqrt{\frac{\sum_{i=1}^{n} |R_i - P_i|^2}{\sum_{i=1}^{n} |R_i|^2}}.$$

We also report the R^2 coefficient of determination and the "error standard deviation" which we define as

$$\sigma := \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left(E - \frac{|R_i - P_i|}{\|R\|_{\bar{\ell}^2}}\right)^2}.$$

We also report the slope of the regression line between true values and the predicted ones.

In Fig. 1 one sees a better fit for a small time $t = 0.05$ which is explained by the fact that the approximation of u by functions from $\mathcal{M}_\alpha^4$ is local because of the Taylor expansions in the Picard iterations (see Sect. 2.1 and Eq. (2.4)). For larger times $t \in \{0.5, 1\}$ as well as different spatial points $x \in \{0.5, 0.95\}$ there seem to be no big statistical difference in accuracy.

In Table 1, we show average relative ℓ^2 error, slope of the regression line, and R^2 statistic for Algorithm 1 applied to models of heights 1, 2, 3 and 4. All experiments are performed 1000 times (i.e. splitting the data randomly into training/test sets) and the average values over these experiments are reported. Table 1 demonstrates that increasing height indeed allows for a better overall prediction. A similar result holds true for the width: additive width smaller than 3 [which corresponds to the third power in the non-linearity in (3.1)] gives on average a worse error.

Remark 3.3 Note that the error for the middle time $t = 0.5$ is slightly worse than for the end time $t = 1$. This could be caused by using a model without initial conditions instead of the full model. Indeed, using the full model as in Definition 3.1 allows to slightly reduce the error for the prediction at $(t, x) = (0.5, 0.5)$ to 7.4% (with the same $n = 4$ and $\alpha = (3, 2, 0)$)

Table 2 Average relative ℓ^2 errors, slopes, and R^2 for off-the-shelf learning algorithms applied to flattened noise with 100,000 points

RFR			SVR			KNN		
Error (%)	Slope	R^2	Error (%)	Slope	R^2	Error (%)	Slope	R^2
42.7	0.04	0.03	37.2	0.23	0.30	44.2	0.12	0.003

Prediction is performed at space–time point $(t, x) = (1, 0.5)$

while making almost no change to the error for the prediction at $(t, x) = (1, 0.5)$ and at $(t, x) = (0.05, 0.5)$.

A heuristic reason why the effect of the fixed initial condition could be ignored for the parabolic equations could be a good local structure and dissipative properties of the heat operator. The advantage of using models with $\mathcal{J} = \emptyset$ for parabolic equations is that such models contain fewer functions, which both improves the speed of the computation and potentially helps with problems of overfitting.

We compare the results from Algorithm 1 with several basic off-the-shelf learning algorithms. To do this, for $(u, \xi) \in U^{\text{obs}}$ we transform all the space–time points of the forcing ξ into a vector (in this case a vector of 100,000 points) and applied support vector regression (SVR), K-nearest neighbours (KNN), and random forest regressions (RFR) to predict the value of $u(t, x)$ for $(t, x) = (1, 0.5) \in [0, T] \times D$. These algorithms were applied with the default settings in the Python `sklearn` library (e.g. SVR was taken with the RBF kernel) and each algorithm was tested on 1000 realisations of the noise with a 700:300 split for training and testing data as before. We give the results in Table 2. None of these algorithms gave better than 35% error. We also subsampled ξ by taking 50, 200 and 1000 evenly sampled space–time points (vs. the full 100,000 points) to avoid over-fitting, but these three choices only increased the error for each regressor. This short comparison demonstrates that the MFV captures information that is lost by treating the noise simply as a large vector.

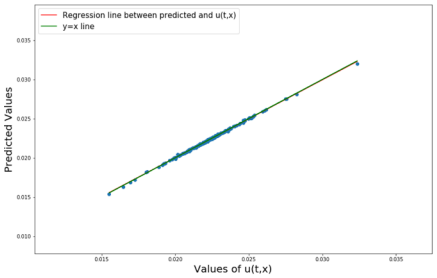
3.1.2 Two-Dimensional Spatial Domain

We consider a similar experiment as in the previous subsection but over a two-dimensional domain. Specifically, we consider the PDE

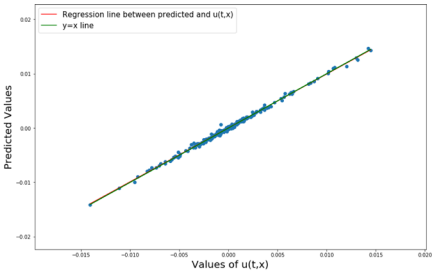
$$\begin{aligned} (\partial_t - \Delta)u &= 3u - u^3 + u\,\xi \quad \text{for}(t, x) \in [0, 1] \times \mathbb{T}^2, \\ u(0, x, y) &= \cos(2\pi(x + y)) + \sin(2\pi(x + y)), \end{aligned} \tag{3.2}$$

where $\mathbb{T}^2 \stackrel{\text{def}}{=} \mathbb{R}^2/\mathbb{Z}^2$ is the two-dimensional torus that we identify with $[0, 1)^2$ as a set (i.e. we consider periodic boundary conditions). We take the forcing ξ now as white in time and coloured in space. The precise definition is $\xi(t, x, y) = \dot{\beta}(t)w(x, y)$ where $\beta(t)$ is Brownian motion, and w is independent of β and normally distributed according to $\mathcal{N}(0, 3^{3/2}(-\Delta + 49I)^{-3})$ as in [43, Sec 4.3]. Here Δ is a periodic Laplacian on $\mathbb{T}^2$ and we take its discrete periodic approximation to generate the data. The reason why we take a coloured noise in space instead of space–time white noise is that the equation (3.2) is singular in two spatial dimensions (see [16]) and does not have a classical solution if the noise is white in both space and time.

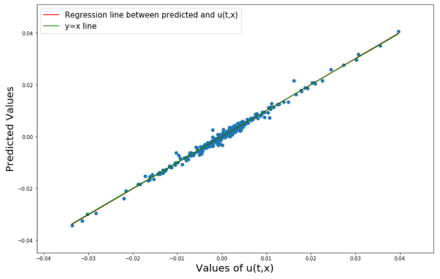
We discretise space into 64×64 points and time into 1000 points. We construct a model with $\mathcal{J} = \{c\}$ a singleton and height $n = 2$ and remaining parameters (m, ℓ, q and degree cut-off) as in Sect. 3.1.1. For the corresponding function u^c in (2.5) we take $u^{(c)} = I_c[u_0] + I[\xi]$



(a) Prediction at $(t, x, y) = (0.05, 0.5, 0.5)$. Relative ℓ^2 error: 0.26%. Slope: 0.995. Error standard deviation: 0.26%. $R^2 = 0.999$



(b) Prediction at $(t, x, y) = (0.5, 0.5, 0.5)$. Relative ℓ^2 error: 6.3%. Slope: 0.994. Error standard deviation: 6.3%. $R^2 = 0.996$.



(c) Prediction at $(t, x, y) = (1.0, 0.5, 0.5)$. Relative ℓ^2 error: 10.9%. Slope: 1.005. Error standard deviation: 10.8%. $R^2 = 0.988$.

Fig. 2 Results of linear regression of solution to (3.2). The x-axis contains values of $u(t, x, y)$, where (t, x, y) is the indicated space–time point, from the test set and the y-axis contains predictions of the linear regression

where I_c and I are as in (2.3). Remark that this definition slightly differs from Definition 3.1; we made this choice to still incorporate the initial condition while keeping the number of functions in $\mathcal{M}_\alpha^n$ relatively small for computational reasons (cf. Remark 3.3). Note though that this implies that the height 0 model already has some non-trivial information (coming from $I[\xi]$).

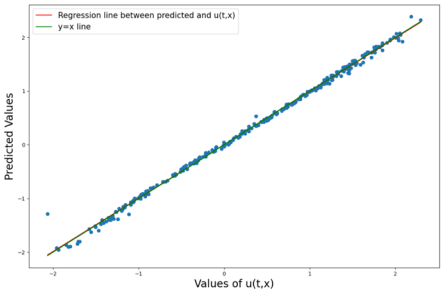
We sampled 1000 realisations of ξ and performed Algorithm 1 as in Sect. 3.1.1 (with 700 training and 300 testing samples). Figure 2 shows the outcome for three space–time points. As in Fig. 1, we see an decrease in accuracy at larger times, which is expected from theory (as explained in Sect. 3.1.1). We also mention that we performed the same experiment with no initial conditions (i.e. with $\mathcal{J} = \emptyset$ as Sect. 3.1.1), but achieved no meaningful predictive power (see Sect. 3.2 for a similar outcome for the wave equation), a feature not encountered in the one-dimensional setting of Sect. 3.1.1.

We furthermore performed the experiment 1000 times, each time resampling the training and testing set, and record the averages of the relative ℓ^2 error, slope of regression line, and R^2 statistic. The results are recorded in Table 3. As in Table 1, we see a sharp rise in predictive power with the height of the model, further demonstrating that non-linearities in the MFV capture important information of the underlying signal. We note that height $n = 2$ here effectively corresponds to height $n = 3$ of Sect. 3.1.1 since we included $I[\xi]$ in $u^{(c)}$.

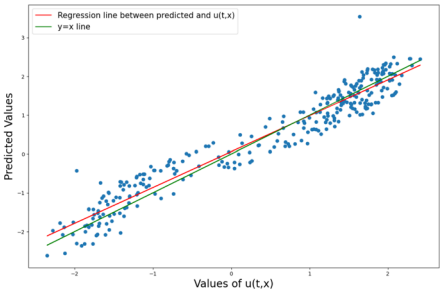
Table 3 Average relative ℓ^2 errors, slopes, and R^2 for linear regression against models of different heights for Eq. (3.2)

Model height	Error (%)	Slope	R^2	Error (%)	Slope	R^2
	$(t, x, y) = (0.05, 0.5, 0.5)$			$(t, x, y) = (0.5, 0.5, 0.5)$		
0	7.0	0.06	0.05	100.3	0.00	0.00
1	1.3	0.97	0.97	46.8	0.79	0.78
2	0.27	0.999	0.999	6.1	0.996	0.996
	$(t, x) = (1, 0.5, 0.5)$					
0	100.1	0.00	0.00			
1	65.2	0.59	0.57			
2	9.8	0.991	0.990			

Prediction is performed at the same space–time points as in Fig. 2



(a) Prediction at $(t, x) = (0.5, 0.5)$. Relative ℓ^2 error: 5.7%. Slope: 0.995. Error standard deviation: 5.6%. $R^2 = 0.997$.



(b) Prediction at $(t, x) = (1, 0.5)$. Relative ℓ^2 error: 25.6%. Slope: 0.92. Error standard deviation: 22.8%. $R^2 = 0.93$.

Fig. 3 Results of linear regression of solution to (3.3) against the functions in model $\mathcal{M}_\alpha^5$ with $\alpha = (3, 1, 0)$, without initial conditions and of degree ≤ 7.5. The x-axis contains values of $u(t, x)$ for realisations of the forcing ξ from the test set and the y-axis contains predictions of the linear regression. Subplots **a**, **b** show predictions at space–time points $(t, x) = (0.5, 0.5)$ and $(t, x) = (1, 0.5)$ respectively

3.1.3 Additive Forcing

We repeat the same experiment for the additive version of the Eq. (3.1) namely:

$$\begin{aligned} &(\partial_t - \Delta)u = 3u - u^3 + \xi \quad \text{for}(t, x) \in [0, 1] \times [0, 1], \\ &u(t, 0) = u(t, 1) \quad \text{(Periodic BC)}, \\ &u(0, x) = x(1 - x). \end{aligned} \tag{3.3}$$

Discretisation of space–time and number of training and test cases is the same as in Sect. 3.1.1. We perform Algorithm 1 using the model $\mathcal{M}_\alpha^5$ from Definition 3.1 with viscosity $\nu = 1$ and $\alpha = (3, 1, 0)$, without initial conditions ($\mathcal{J} = \emptyset$), and with degree ≤ 7.5, which gives 58 functions.[8] Note that since multiplicative width is 1 this reduces the number of functions in the model compared to the multiplicative case of Sect. 3.1.1, which allows us to take a larger height and upper bound for the degree. Figure 3 shows the results for space–time points $(t, x) = (0.5, 0.5)$ and $(t, x) = (1, 0.5)$. One can see that the additive equation

[8] See Remark 2.5 for a motivations behind taking these particular widths.

exhibits a worse prediction for long times compared to the multiplicative equation (Fig. 1, Table 1) but a slightly better prediction for short times (for $t = 0.05$ error is even better: 0.1% in comparison to $\approx 5\%$ in the multiplicative case).

The prediction error rises in this additive case as t increases. This is expected by a similar reason as mentioned in Sect. 3.1.1, which is the Taylor expansions in the Picard iterations approximating u (see (2.4)). It would be of interest to extend Algorithm 1 to decrease this error. A potential way to do this is to generalise and combine Algorithms 1 and 2 and compute models on subintervals to learn the 'flow' of the equation, i.e. build models from the initial condition and the forcing over subintervals of $[0, T]$, use this model to predict the solution over subintervals, and chain the predictions together. (See also the discussion at the end of Sect. 2.3.1.) We leave this generalisation for a future work.

3.2 Wave Equation with Forcing

We will now consider a wave equation taking $\mathcal{L} = \partial_t^2 - \Delta$ in (2.1) and predict solutions of the following non-linear wave equation

$$\begin{aligned} &(\partial_t^2 - \Delta)u = \cos(\pi\, u) + u^2 + u\,\xi \quad \text{for}(t, x) \in [0, 1] \times [0, 1], \\ &u(t, 0) = u(t, 1) \quad \text{(Periodic BC)}, \\ &u(0, x) = \sin(2\pi\, x), \\ &\partial_t u(0, x) = x(1 - x), \end{aligned} \tag{3.4}$$

where ξ is a space–time forcing which we again take as a realisation of white noise. We will compare Algorithm 1 with both models with and without initial conditions. Discretisation of space–time and number of training and test cases is the same as in Sect. 3.1.1.

Note that, in the general case, the level zero of the full model $\mathcal{M}_\alpha^0$ for the wave equation should not only include the contribution of the initial condition u_0 but also the contribution of the initial speed $\partial_t u(0, x) = v_0$. This leads to the following definition.

Definition 3.4 Consider an initial condition $u_0 : D \to \mathbb{R}$, an initial speed $v_0 \colon D \to \mathbb{R}$, and a forcing $\xi : [0, T] \times D \to \mathbb{R}$, as well as $n, m, \ell \geq 1$, $q \leq 1$ and $\nu > 0$. Let $\alpha = (m, \ell, q)$. The model $\mathcal{M}_\alpha^n$ for the wave equation with propagation speed $\sqrt{\nu}$ is constructed by taking $\mathcal{J} = \{c, s\}$ with $u^{(c)} = I_c[u_0]$, $u^{(s)} = I_s[v_0]$ where

$$\begin{cases} (\partial_t^2 - \nu\Delta) I_c[u_0] &= 0 \\ I_c[u_0](0, x) &= u_0(x), \\ \partial_t I_c[u_0](0, x) &= 0. \end{cases} \qquad \begin{cases} (\partial_t^2 - \nu\Delta) I_s[v_0] &= 0 \\ I_s[v_0](0, x) &= 0, \\ \partial_t I_s[v_0](0, x) &= v_0(x). \end{cases}$$

Moreover, for functions $f : [0, T] \times D \to \mathbb{R}$ the operator $I[f]$ is defined to be the solution to a wave equation

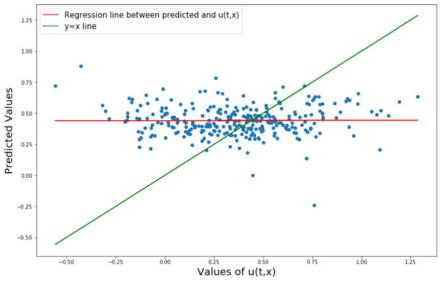
$$(\partial_t^2 - \nu\Delta) I[f] = f,$$

with $I[f](0, x) = \partial_t I[f](0, x) = 0$.

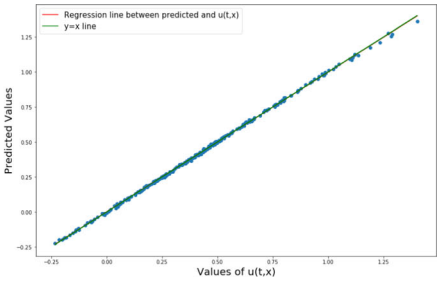
Boundary conditions for the above equation are taken to be the same as boundary conditions for the underlying wave equation (in this case periodic).

For this experiment we choose $n = 4$, $\alpha = (m, \ell, q) = (2, 2, 0)$, and $\nu = 1$, and impose the degree to satisfy $\deg\xi = -1.5$, $\deg u_0 = \deg v_0 = -0.5$ and $\beta = 1.5$ in (2.10). We choose only functions of degree ≤ 1.5 which gives 60 functions in $\mathcal{M}_\alpha^4$.

Figure 4 shows the importance of using the full model in the case of the wave equation,



(a) Prediction at $(t, x) = (1, 0.5)$ for model with $\mathcal{J} = \emptyset$. Relative ℓ^2 error: 72.1%. Slope: 0.003. Error standard deviation: 50.7%. $R^2 = -0.21$.



(b) Prediction at $(t, x) = (1, 0.5)$ for model with $\mathcal{J} = \{c, s\}$. Relative ℓ^2 error: 1.3%. Slope: 0.998. Error standard deviation: 0.9%. $R^2 = 0.999$.

Fig. 4 Results of linear regression of solution to (3.4) using functions from models $\mathcal{M}_\alpha^4$ with $\alpha = (2, 2, 0)$ of degree ≤ 1.5 without and with initial conditions. The x-axis contains values of $u(t, x)$ for realisations of the forcing ξ from the test set and the y-axis contains predictions of the linear regression. Subplots **a**, **b** show predictions at space–time point $(t, x) = (1, 0.5)$ for models with $\mathcal{J} = \emptyset$ and $\mathcal{J} = \{c, s\}$ respectively

Table 4 Average relative ℓ^2 errors, slopes, and R^2 for prediction at space–time point $(t, x) = (1, 0.5)$ for different heights of the model

Model height	With initial speed			Without initial speed		
	Error (%)	Slope	R^2	Error (%)	Slope	R^2
1	59.8	0.04	0.03	59.8	0.03	0.03
2	12.8	0.96	0.96	13.8	0.95	0.95
3	2.1	0.999	0.999	5.0	0.994	0.993
4	1.4	0.999	0.999	4.3	0.995	0.995

First column involves models using $\mathcal{J} = \{c, s\}$ and the second column involves models using $\mathcal{J} = \{c\}$ only

i.e. the contribution of the initial condition (even fixed and deterministic) can't be ignored.[9] The model constructed with $\mathcal{J} = \emptyset$ in the case of the wave equation gives absolutely no predictability contrary to the parabolic case (see Remark 3.3) because the wave operator is not dissipative contrary to the heat operator.

Average relative ℓ^2 errors corresponding to different heights of the model with and without initial speed are presented in the Table 4 for 1000 repeated experiments. Table 4 further shows the importance of including the contributions of both the initial condition and the initial speed in the model when predicting the wave equation.

3.3 Burgers' Equation

In this subsection, we aim to predict solutions to the following Burgers' equation with no forcing

$$(\partial_t - 0.2\Delta)u = -u\partial_x u, \quad (t, x) \in [0, 10] \times [-8, 8]$$
$$u(t, -8) = u(t, 8) \quad \text{(Periodic BC)},$$

[9] This parallels the necessity of including the initial condition in an analogue of the model in [14] where the authors solve a non-linear singular stochastic wave equation in 3 dimensions.

Table 5 Performance of Algorithm 2 on predicting solutions to (3.5) with no noise and true viscosity $\nu = 0.2$, with estimated viscosity (EV), and with 1% noise and 3% noise on observed data

	Algorithm 2				
	AE	AER	TER	ASD	SDR
No noise	1.1%	0.8–1.6%	0.06–9.4%	1.1%	0.8–1.9%
EV	1.1%	0.8–1.6%	0.06–9.2%	1.1%	0.7–1.9%
1% noise	8.8%	5.8–12.4%	0.4–50.0%	8.0%	4.9–11.1%
3% noise	13.7%	8.7–13.7%	1.8–68.3%	8.7%	4.8–12.8%
	PDE-FIND				
No noise	0.9%	0.5–2.3%	0.2–10.5%	0.6%	0.2–2.2%
1% noise	7.3%	4.6–14.2%	0.9–32.7%	5.5%	3.3–8.9%
3% noise	19.1%	13.9–25.1%	3.2–77.1%	12.5%	7.8–18.4%

Comparison is given with PDE-FIND on observations with no noise, 1%, and 3% noise (see Sect. 3.3.1). AE, average relative ℓ^2 errors over all experiments and test cases; AER, average error range over all experiments; TER, total error range over all experiments and test cases; ASD, average standard deviation over all experiments; SDR, standard deviation range over all experiments

$$u(0,x) = \sum_{k=-10}^{10} \frac{a_k}{1+|k|^2} \sin\left(\lambda^{-1}\pi k x\right) \tag{3.5}$$

from the knowledge of the initial condition u_0. That is, given only the initial condition $u_0 \colon [-8,8] \to \mathbb{R}$, our goal is to reconstruction the entire function $u \colon [0,10] \times [-8.8] \to \mathbb{R}$ without explicitly solving the PDE (3.5), i.e., we wish to learn the map $u_0 \mapsto u$. This experiment is partly inspired by [34]. The above equation satisfies the time homogeneous flow property that motivates Algorithm 2, which we use in the experiments below.

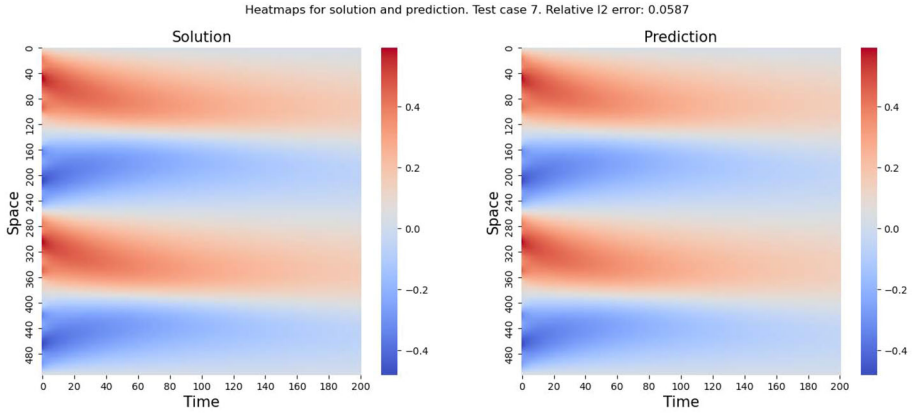
Above, $(a_k)_{k=-10,\dots,10}$ are sampled as independent and identically distributed (i.i.d.) standard normal random variables and λ is a scaling parameter. We sample 120 such initial conditions with scaling $\lambda = 8, 4, 2$ (40 initial conditions for each scaling), which corresponds to u_0 having respectively one, two and four cycles. We then randomly subdivide these initial conditions into training and test sets of sizes 100 and 20 respectively.

To discretise time, we take 201 evenly spaced points $\mathcal{O}_T \subset [0,10]$ ($N = 200$, $\delta = 0.05$ in the notation of Sect. 2.3.2). To approximate the spatial domain, we take 512 evenly spaced points $\mathcal{O}_X \subset D = [-8,8]$. Solutions to the equation, however, we generated using a finer grid (2001 time points and 512 space points).

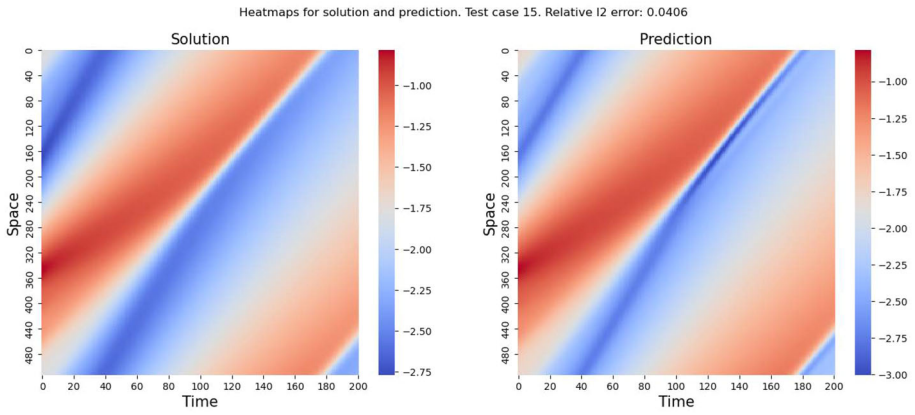
There is no forcing ξ in the equation so $\ell = 0$. We construct the models using Definition 3.1 with viscosity $\nu = 0.2$. In constructing the model, we discretise time with the finer grid, so the relevant space–time domain in continuum is $[0,\delta] \times D$ which we discretise to $\{0, \delta/10 \dots, 9\delta/10, \delta\} \times \mathcal{O}_X$. In particular, the operator I_c takes as input a function $u_0 \colon \mathcal{O}_X \to \mathbb{R}$ and outputs a function $I_c[u_0] \colon \{0, \delta/10 \dots, 9\delta/10, \delta\} \times \mathcal{O}_X \to \mathbb{R}$.

We choose height $n = 3$, additive width $m = 2$, and differentiation order $q = 1$ for the model, i.e. $\alpha = (2,0,1)$. We assign for the degree $\deg u_0 = -1.5$ and $\beta = 2$ in (2.10) and only functions f_τ with degree $\deg\tau \le 2.5$ are considered. This gives 20 functions of $\deg\tau \le 2.5$ instead of the original 91 functions in $\mathcal{M}_\alpha^3$. Using this degree cutoff speeds up the fitting of the linear regression by around 20 times in addition to a faster computation of the model.

As the number of training and testing cases (100 and 20) is relatively small, we repeated the above experiment 10 times. In the first row of Table 5 we record the performance of



(a) Relative ℓ^2 error: 5.9%.



(b) Relative ℓ^2 error: 4.1%.

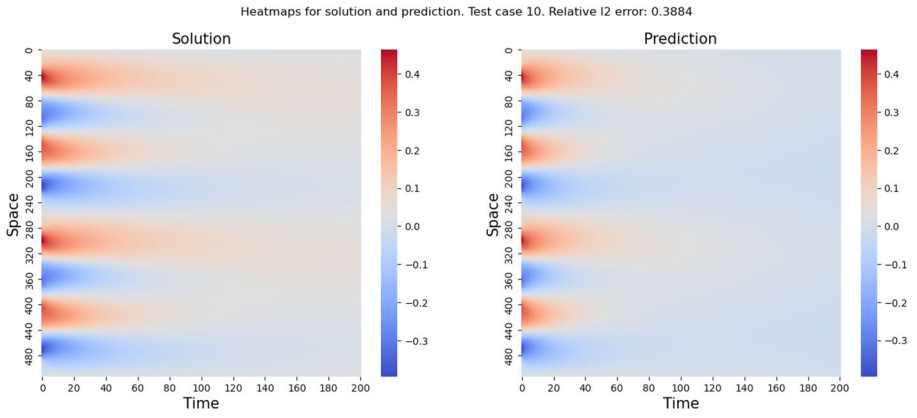
Fig. 5 Heat-maps for the solutions of (3.5) (left) and predictions for two test cases (right) using Algorithm 2 with functions from the model $\mathcal{M}_\alpha^3$ with $\alpha = (2, 0, 1)$ of degree ≤ 2.5. The values of λ are 4 for subfigure (**a**) and 8 for (**b**)

Algorithm 2, namely the averages, ranges, and standard deviations of the relative ℓ^2 error over the 10 experiments with 20 test cases each. Here the relative ℓ^2 error for one test case with true solution R and predicted solution P is defined as

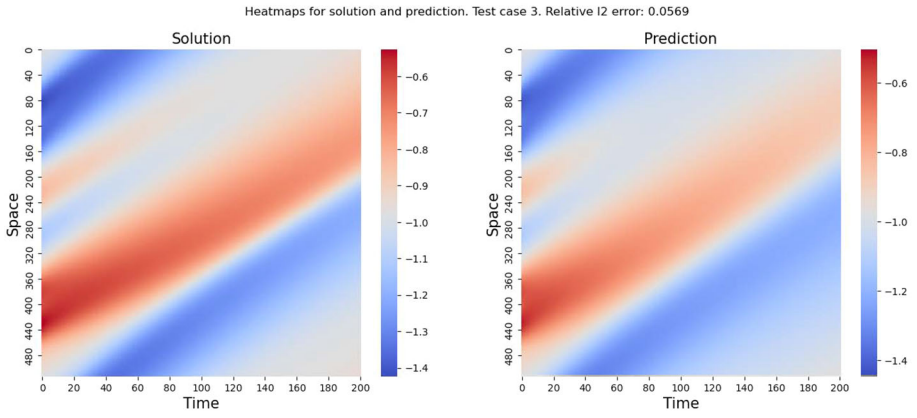
$$E = \frac{\|R - P\|_{\tilde{\ell}^2}}{\|R\|_{\tilde{\ell}^2}}, \qquad \|R\|_{\tilde{\ell}^2}^2 := \frac{1}{201 \times 512} \sum_{(t,x)} |R(t,x)|^2$$

(the sum is over the observed grid points $\mathcal{O}_T \times \mathcal{O}_X \subset [0, 10] \times [-8, 8]$).

Figure 5 shows the heat-maps for the true and predicted solutions drawn from two test cases with greater than average error (heat-maps for test cases with error close to the average error appeared indistinguishable to the naked eye; even for Fig. 5a, where the error of 5.9% is above the average, the two solutions appear similar).



(a) Relative ℓ^2 error: 38.8%.



(b) Relative ℓ^2 error: 5.7%.

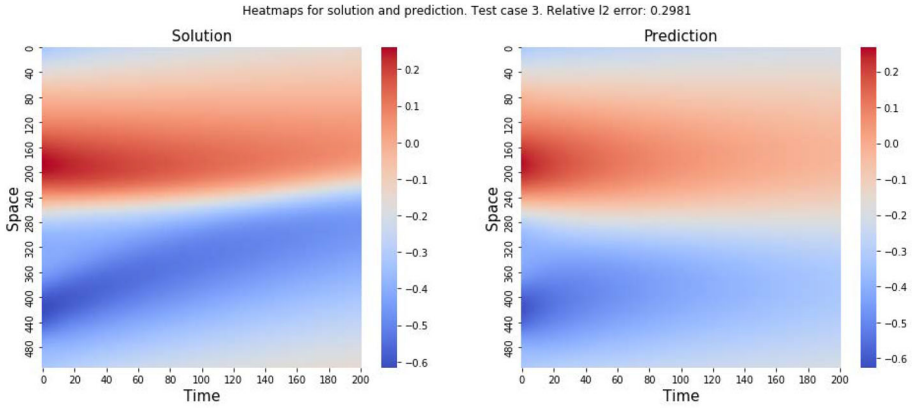
Fig. 6 Heat-maps for the solutions of (3.5) (left) and predictions for two test cases (right) using Algorithm 2 with 1% error in observed samples. The values of λ are 2 for subfigure (**a**) and 8 for (**b**)

We furthermore tested Algorithm 2 on noisy data. We added a 1% error (resp. 3%) to the observed data in the following way. Instead of observing the solution u of (3.5), we observe

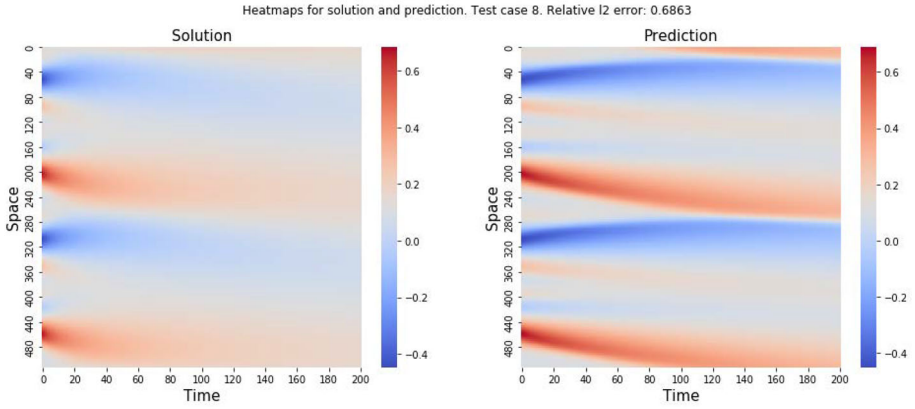
$$\tilde{u}(t,x) = u(t,x) + \varepsilon(t,x)\|u\|_{\tilde{l}^1}, \tag{3.6}$$

where $\|u\|_{\tilde{l}^1} = \frac{1}{201\times 512}\sum_{(t,x)}|u(t,x)|$ (the sum is over the observed grid points $\mathcal{O}_T \times \mathcal{O}_X \subset [0,10]\times[-8,8]$) and $\varepsilon(t,x)$ are i.i.d. normal random variables with zero mean and standard deviation 0.01 (resp. 0.03) for each (t,x) for the training data and for $(t,x) = (0,x)$ for the test data. The corresponding errors, over 10 experiments, are presented in Table 5. In Fig. 6 we give two examples of heat-maps with varying relative ℓ^2 error for 1% noise and in Fig. 7a we give an example with 3% noise.

Finally, we ran Algorithm 2 using models in which the viscosity parameter ν in Definition 3.1 is estimated from the data (the true value being $\nu = 0.2$). We estimate ν by simply linearly regressing the discrete time derivative of u against the discrete second derivative in space for $u \in U^{\text{obs}}$. To be more precise, we use ordinary least squares to determine the best



(a) Algorithm 2, 3% noise, relative ℓ^2 error: 29.8%.



(b) PDE-FIND, 3% noise, relative ℓ^2 error: 68.6%.

Fig. 7 Heat-maps for the solutions of (3.5) (left) and predictions for two test cases (right) with 3% noise on observed data. Subfigure **a** is for Algorithm 2 and subfigure **b** is for PDE-FIND. The values of λ are 8 for subfigure (**a**) and 4 for (**b**)

Table 6 Estimated viscosity $\tilde{\nu}$ via linear regression over 10 experiments

Average	Range	Standard deviation
0.177	0.171–0.182	0.003

$\tilde{\nu}$ that fits

$$(u_{t_1}(x) - u_{t_0}(x))/\delta = \epsilon + \tilde{\nu}\partial_x^2 u_{t_0}(x), \quad u \in U^{\text{obs}}, \ x \in D,$$

where ∂_x^2 is computed using central finite difference. Such an estimate does not require any knowledge of the non-linearity. (One could further use cross-validation to find a better estimate for ν from the interval $[\tilde{\nu} - a, \tilde{\nu} + a]$ for some $a > 0$, although we did not do this.) The average, range, and standard deviation of the estimated viscosity over the 10 experiments is recorded in Table 6.

Table 7 Estimated parameters a and $-b$ in (3.7) via PDE-FIND

	a			$-b$		
	Average	Range	SD	Average	Range	SD
No noise	0.212	0.203–0.246	0.012	0.984	0.979–0.987	0.002
1% noise	0.115	0.085–0.139	0.016	0.902	0.742–0.952	0.058
3% noise	0.027	0.021–0.038	0.005	0.733	0.640–0.796	0.045

The true values are $a = 0.2$ and $-b = 1$

We record the errors in Table 5 for Algorithm 2 with these estimated viscosities. We find that the results are essentially the same as those with the correct viscosity $\nu = 0.2$ (and in fact have tiny improvements over the latter).

3.3.1 Comparison with PDE-FIND Algorithm

We use a version of PDE-FIND algorithm from [38] to learn the non-linearity first instead of learning the solution. We use linear regression to find the best coefficients a, b such that

$$\partial_t u(t_k, x) = a\partial_x^2 u(t_k, x) + bu(t_k, x)\partial_x u(t_k, x), \tag{3.7}$$

for $u \in U^{\text{obs}}$, $k = 0, \dots N-1$, and $x \in D$, where $\partial_t u(t_k, x) := \frac{u(t_{k+1},x)-u(t_k,x)}{\delta}$ is a discrete time derivative, ∂_x is discrete space derivative and $x \in D$ are the observed points.

We then use finite difference method with the estimated (a, b) (estimating these coefficients separately for each experiment) starting from initial conditions from U^{pr} in order to construct the predicted solution on the full domain $[0, 10] \times D$. Note though that in this finite difference we can discretise time on a finer grid. In fact, we take 2001 points on $[0, 10]$ which is the same number of time points that is used to construct solution to (3.5). We cannot, however, discretise the spatial domain $[-8, 8]$ to a finer grid than 512 points because these are the only points observed for the initial conditions from U^{pr}.

The estimated coefficients a, b are recorded in Table 7. The resulting errors for the predicted solutions after repeating the experiment 10 times with 20 test cases as before are recorded in Table 5. We notice that the performance of Algorithm 2 (with and without estimated viscosity) is similar to that of PDE-FIND, although Algorithm 2 yields a slightly larger average error, but a slightly lower maximum average error and total error.

We furthermore performed the same experiments but with 1% noise and 3% noise on observed samples as for Algorithm 2. Here, instead of direct linear regression, we follow the proposal in [38] and perform polynomial interpolation: for each space–time point z, we fit a polynomial of degree 4 that best matches the observed function in a neighbourhood of radius 20 points around z, and then estimate a, b in (3.7) by taking derivatives of these polynomials and applying linear regression. This is made in order to avoid taking explicit derivatives of $\tilde{u}$ via the finite difference method since the noisy data is not differentiable. The resulting errors and estimates for a, b are recorded in Tables 5 and 7 respectively. On 1% noise, the two methods are again comparable (with PDE-FIND demonstrating a slightly lower average error).

However, with 3% noise, we found that there is a noticeable difference. First, the estimated viscosity a in Table 7 is between 0.02 and 0.04, which is significantly lower than the true value 0.2. This caused the predicted solution to blow up on some test cases (due to numerical instability in our finite difference method): in each of the 10 experiments, between 0 and 7

of the 20 test cases blew up (the two extreme values were attained only for one experiment each, and the most common number of blow-ups was 1). Figure 7b shows heat-maps for a non-blow-up test case with 3% noise using PDE-FIND, wherein one can see the effect of the low estimated viscosity. In comparison, no test cases for Algorithm 2 blew up.

In Table 5 we only report errors from test cases where PDE-FIND did *not* blow up—the errors are expected to be even larger if all test cases were included by solving the associated equation with a more sophisticated numerical scheme. Even after removing the test cases for which PDE-FIND blew up, we see a mild advantage of Algorithm 2 over PDE-FIND with polynomial interpolation.

Finally, the reader may wonder if it is fair to compare Algorithm 2 to PDE-FIND given that we input into Algorithm 2 the true viscosity 0.2, while PDE-FIND is required to estimate it. We point out, however, that Algorithm 2 has no knowledge of the non-linearity $u\partial_x u$ in (3.5) (though the parameters m, q are chosen with the motivation that the non-linearity is at most quadratic with $\partial_x u$ possibly appearing), while in our implementation of PDE-FIND we do input $u\partial_x u$ as the only possible non-linearity. Furthermore, on noiseless data, the viscosity estimated from the data gives results for Algorithm 2 that are comparable to PDE-FIND (see Table 5). We also point out that Algorithm 2 was approximatively 20 times faster to run with a fast Fourier transform method of computing models than PDE-FIND with polynomial interpolation.

4 Summary and Discussion

To summarise, we proposed a new *model feature vector* (MFV) of a space–time signal that extends to multi-dimensional space the notion of a path signature. We further proposed two regression algorithms, which reveal that MFVs may contain important information about the underlying signal.

We applied Algorithm 1 to both parabolic and hyperbolic equations with forcing and Algorithm 2 to Burgers' equation with varying initial conditions. We did an elementary comparison of our algorithms with other methods. We compared the performance of Algorithm 1 for the parabolic equation with multiplicative forcing against several off-the-shelf methods and found a large advantage in favour of Algorithm 1. We further compared Algorithm 2 for the Burgers' equation against a version of PDE-FIND [38] (see Sect. 3.3.1). The two methods were comparable (with PDE-FIND showing a minor advantage) on noiseless and small noise data, while Algorithm 2 showed an advantage over PDE-FIND with larger noise data. We believe the success of Algorithm 2 in this experiment is due to the smoothing properties of the heat operator, which provides considerable robustness.

In terms of the hyperparameters, the experiments with Algorithm 1 in Sects. 3.1 and 3.2 show that increasing the height of the model gives better predictability. We also found in Sect. 3.3 that one can effectively estimate the viscosity parameter $\nu > 0$ at no expense in the error. These experiments demonstrate a potential for the use of MFVs as features for learning PDEs. A more systematic comparison of our algorithms with other methods as well as analysis of the effect of hyperparameters is left for future work.

We conclude by discussing several other directions in which this work could be extended.

- *Beyond PDEs.* An important next step is to investigate the use of models as features in learning algorithms in contexts beyond PDEs. We believe that natural directions to investigate include analysis of meteorological data [9, 42], image and remote sensing

recognition [26, 46], and applications to fluid dynamics [4, 28]. Such extensions would parallel the current use of signatures in data science well beyond the scope of ODEs.

- *Universality.* It would be of interest to understand universality properties of models, i.e. in what sense and under which conditions can one approximate general functions of the input $(\{u^{(i)}\}_{i\in\mathcal{J}}, \xi)$ with *linear* functions of the model. Beyond their importance in machine learning, such universality properties are of deep mathematical interest; a celebrated result is that linear functions of the signature (see Definition 2.3) approximate, uniformly on compact sets, continuous function of rough paths modulo *tree-like equivalence* [2, 19].
- *Further learning algorithms.* It will be important to explore the utility of 'model features' when combined with learning algorithms beyond linear regression (the only tool used in this article), such as with neural networks and random forests. Similarly, it would be important to *kernelise* the model feature vector efficiently. This would allow for use of popular kernel learning methods, such as support vector machines, and of the maximum mean discrepancy (MMD) of [13] to compare samples drawn from different distributions. An MMD from the kernelised signature map was used in [11] to define a metric on the laws of stochastic process indexed by time, and fast signature kernelisation algorithms were introduced in [25]; extending these results to models would be of significant interest.
- *Higher dimensions.* In order to apply the ideas in this paper to data in high dimensional spaces, it would be important to improve the computation of models. It took[10] between 0.2 to 0.5 s to compute one model in Sects. 3.1 and 3.2, and approximately 90 s to perform one run of Algorithm 2 in Sect. 3.3. The computation time in higher spatial dimensions would be significantly longer. In this direction, there are a number of works aiming to solve high dimensional PDEs with learning algorithms, such as [18, 44]. Since, with the choice of operator I in our experiments, elements of the models are solutions to special PDEs, it would be interesting to see if these methods could make it feasible to compute the model features in high dimensional spaces.
- *Operator I hyperparameter.* The operator I in the definition of a model is a hyperparameter which needs to be chosen from a very large space (the infinite-dimensional space of linear operators). In our experiments, we mostly used knowledge of the *linear part* of the PDE (heat or wave operators) to choose I. However, if the PDE is completely unknown, or if the output u does not come from a PDE at all, then one would need a systematic way to choose this hyperparameter. The same applies to the other hyperparameters, such as n, m, ℓ, q, but these take values in a smaller space for which standard hyperparameter tuning (e.g. cross-validation, or sparse linear regression similar to PDE-FIND [38]) is feasible. Note that the problem of choosing I does not arise in the context of signatures simply because one hardcodes I as convolution with the Heaviside step function $J(t) = \mathbf{1}_{t>0}$. We could of course likewise hardcode I, e.g., as the inverse of the heat operator (2.3), but if one believes the output u should behave like the solution to a wave equation, this will likely yield poor performance. How to choose I in a general context is therefore an important theoretical and practical question.

Acknowledgements The authors would like to thank the anonymous referees for their thorough reading of the paper and suggestions for improvements.

Funding AG and HW were supported by the Leverhulme Trust through a Philip Leverhulme Prize during the writing of this article. HW was also supported by the Royal Society through the University Research Fellowship UF140187.

[10] On a laptop with 4 Cores (1.4 GHz) and 16 GB memory.

Data Availibility All code and data are publicly available at https://github.com/andrisger/Feature-Engineering-with-Regularity-Structures.git.

Declarations

Conflict of interest The authors have no competing interests to declare.

References

1. Arribas, I.P., Goodwin, G.M., Geddes, J.R., Lyons, T., Saunders, K.E.A.: A signature-based machine learning model for distinguishing bipolar disorder and borderline personality disorder. Transl. Psychiatry **8**, 274 (2018). https://doi.org/10.1038/s41398-018-0334-0
2. Boedihardjo, H., Geng, X., Lyons, T., Yang, D.: The signature of a rough path: uniqueness. Adv. Math. **293**, 720–737 (2016). https://doi.org/10.1016/j.aim.2016.02.011
3. Bruned, Y., Hairer, M., Zambotti, L.: Algebraic renormalisation of regularity structures. Invent. Math. **215**(3), 1039–1156 (2019). https://doi.org/10.1007/s00222-018-0841-x. arXiv:1610.08468
4. Brunton, S.L., Noack, B.R., Koumoutsakos, P.: Machine learning for fluid mechanics. Ann. Rev. Fluid Mech. **52**(1), 477–508 (2020). https://doi.org/10.1146/annurev-fluid-010719-060214
5. Bar-Sinai, Y., Hoyer, S., Hickey, J., Brenner, M.P.: Learning data-driven discretizations for partial differential equations. Proc. Natl. Acad. Sci. USA **116**(31), 15344–15349 (2019). https://doi.org/10.1073/pnas.1814058116
6. Chen, K.-T.: Integration of paths—a faithful representation of paths by non-commutative formal power series. Trans. Am. Math. Soc. **89**, 395–407 (1958). https://doi.org/10.2307/1993193
7. Chevyrev, I., Kormilitzin, A.: A primer on the signature method in machine learning (2016). arXiv:1603.03788
8. Chevyrev, I., Lyons, T.: Characteristic functions of measures on geometric rough paths. Ann. Probab. **446**, 4049–4082 (2016). https://doi.org/10.1214/15-AOP1068
9. Chen, G., Li, S., Knibbs, L.D., Hamm, N., Cao, W., Li, T., Guo, J., Ren, H., Abramson, M.J., Guo, Y.: A machine learning method to estimate pm2.5 concentrations across China with remote sensing, meteorological and land use information. Sci. Total Environ. **636**, 52–60 (2018). https://doi.org/10.1016/j.scitotenv.2018.04.251
10. Chevyrev, I., Nanda, V., Oberhauser, H.: Persistence paths and signature features in topological data analysis. IEEE Trans. Pattern Anal. Mach. Intell. **42**(1), 192–202 (2020). https://doi.org/10.1109/TPAMI.2018.2885516
11. Chevyrev, I., Oberhauser, H.: Signature moments to characterize laws of stochastic processes. J. Mach. Learn. Res. **23**(176), 1–42 (2022). arXiv:1810.10971
12. Prato, G.D., Zabczyk, J.: Stochastic equations in infinite dimensions. In: Encyclopedia of Mathematics and its Applications, vol. 152, 2nd edn., p. xviii+493. Cambridge University Press, Cambridge (2014). https://doi.org/10.1017/CBO9781107295513
13. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. J. Mach. Learn. Res. **13**, 723–773 (2012)
14. Gubinelli, M., Koch, H., Oh, T.: Paracontrolled approach to the three-dimensional stochastic nonlinear wave equation with quadratic nonlinearity. To appear in J. Eur. Math. Soc. (2021). arXiv:1811.07808
15. Graham, B.: Sparse arrays of signatures for online character recognition (2013). arXiv:1308.0371
16. Hairer, M.: A theory of regularity structures. Invent. Math. **198**(2), 269–504 (2014). https://doi.org/10.1007/s00222-014-0505-4. arXiv:1303.5113

17. Harrell, F.E.: Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis. Springer Series in Statistics, 2nd edn., p. xxv+582. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19425-7
18. Han, J., Jentzen, A.W.E.: Solving high-dimensional partial differential equations using deep learning. Proc. Natl. Acad. Sci. USA **115**(34), 8505–8510 (2018). https://doi.org/10.1073/pnas.1718942115
19. Hambly, B., Lyons, T.: Uniqueness for the signature of a path of bounded variation and the reduced path group. Ann. Math. (2) **1711**, 109–167 (2010). https://doi.org/10.4007/annals.2010.171.109
20. Hu, P., Meng, Q., Chen, B., Gong, S., Wang, Y., Chen, W., Zhu, R., Ma, Z.-M., Liu, T.-Y.: Neural operator with regularity structure for modeling dynamics driven by SPDEs (2022). arXiv e-prints arXiv:2204.06255
21. Huang, J., Wang, H., Zhou, T.: An augmented Lagrangian deep learning method for variational problems with essential boundary conditions. Commun. Comput. Phys. **31**(3), 966–986 (2022). https://doi.org/10.4208/cicp.oa-2021-0176
22. Jagtap, A.D., Karniadakis, G.E.: Extended physics-informed neural networks (XPINNs): a generalized space–time domain decomposition based deep learning framework for nonlinear partial differential equations. Commun. Comput. Phys. **28**(5), 2002–2041 (2020). https://doi.org/10.4208/cicp.oa-2020-0164
23. Kidger, P., Bonnier, P., Perez Arribas, I., Salvi, C., Lyons. T.: Deep signature transforms. In: Advances in Neural Information Processing Systems, vol. 32, pp. 3105–3115. Curran Associates, Inc. (2019)
24. Kalsi, J., Lyons, T., Arribas, I.P.: Optimal execution with rough path signatures. SIAM J. Financ. Math. **11**(2), 470–493 (2020). https://doi.org/10.1137/19M1259778
25. Kiraly, F.J., Oberhauser, H.: Kernels for sequentially ordered data. J. Mach. Learn. Res. **20**(31), 1–45 (2019)
26. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical Report, Department of Computer Science, University of Toronto (2009)
27. Li, H., Khoo, Y., Ren, Y., Yin, L.: A semigroup method for high dimensional committor functions based on neural network. In: Bruna, J., Hesthaven, J., Zdeborova, L. (eds.) Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference, Proceedings of Machine Learning Research, vol. 145, pp. 598–618. PMLR (2022)
28. Ling, J., Kurzawski, A., Templeton, J.: Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J. Fluid Mech. **807**, 155–166 (2016). https://doi.org/10.1017/jfm.2016.615
29. Liao, Y., Ming, P.: Deep Nitsche method: deep Ritz method with essential boundary conditions. Commun. Comput. Phys. **295**, 1365–1384 (2021). https://doi.org/10.4208/cicp.oa-2020-0219
30. Lyons, T., Nejad, S., Arribas, I.P.: Numerical method for model-free pricing of exotic derivatives in discrete time using rough path signatures. Appl. Math. Finance **26**(6), 583–597 (2019). https://doi.org/10.1080/1350486X.2020.1726784
31. Lord, G.J., Powell, C.E., Shardlow, T.: An introduction to computational stochastic PDEs. Cambridge Texts in Applied Mathematics, p. xii+503. Cambridge University Press, New York (2014). https://doi.org/10.1017/CBO9781139017329
32. Lyons, T.J.: Differential equations driven by rough signals. Rev. Mat. Iberoam. **14**(2), 215–310 (1998). https://doi.org/10.4171/RMI/240
33. Li, C., Zhang, X., Jin, L.: LPSNet: A novel log path signature feature based hand gesture recognition framework. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 631–639 (2017)
34. Malik, S., Anwar, U., Ahmed, A., Aghasi, A.: Learning to solve differential equations across initial conditions. In: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations (2020)
35. Morrill, J., Kormilitzin, A., Nevado-Holgado, A., Swaminathan, S., Howison, S., Lyons, T.: The signature-based model for early detection of sepsis from electronic health records in the intensive care unit. In: 2019 Computing in Cardiology (CinC), pp. 1–4 (2019)
36. Magill, M., Qureshi, F., de Haan, H.: Neural networks trained to solve differential equations learn general representations. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc. (2018)
37. Papavasiliou, A., Ladroue, C.: Parameter estimation for rough differential equations. Ann. Stat. **39**(4), 2047–2073 (2011). https://doi.org/10.1214/11-AOS893
38. Rudy, S.H., Brunton, S.L., Proctor, J.L., Kutz, J.N.: Data-driven discovery of partial differential equations. Sci. Adv. **3**(4), e1602614 (2017). https://doi.org/10.1126/sciadv.1602614
39. Rowley, C.W., Mezić, I., Bagheri, S., Schlatter, P., Henningson, D.S.: Spectral analysis of nonlinear flows. J. Fluid Mech. **641**, 115–127 (2009). https://doi.org/10.1017/S0022112009992059

40. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019). https://doi.org/10.1016/j.jcp.2018.10.045
41. Schmid, P.J.: Dynamic mode decomposition of numerical and experimental data. J. Fluid Mech. **656**, 5–28 (2010). https://doi.org/10.1017/S0022112010001217
42. Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., Woo, W.-C.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 1, pp. 802–810, NIPS'15. MIT Press, Cambridge (2015)
43. Salvi, C., Lemercier, M., Gerasimovics, A.: Neural stochastic PDEs: resolution-invariant learning of continuous spatiotemporal dynamics. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems, vol. 35, pp. 1333–1344. Curran Associates, Inc. (2022)
44. Sirignano, J., Spiliopoulos, K.: DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018). https://doi.org/10.1016/j.jcp.2018.08.029
45. Xie, Z., Sun, Z., Jin, L., Ni, H., Lyons, T.: Learning spatial-semantic context with fully convolutional recurrent network for online handwritten Chinese text recognition. IEEE Trans. Pattern Anal. Mach. Intell. **40**(8), 1903–1917 (2018). https://doi.org/10.1109/TPAMI.2017.2732978
46. Zou, Q., Ni, L., Zhang, T., Wang, Q.: Deep learning based feature selection for remote sensing scene classification. IEEE Geosci. Remote Sens. Lett. **12**(11), 2321–2325 (2015)
47. Zhang, Z., Wang, Y., Jimack, P.K., Wang, H.: Meshingnet: a new mesh generation method based on deep learning. In: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (eds.) Computational Science—ICCS 2020, pp. 186–198. Springer, Cham (2020)