THAT'S OP: LEARNING OF FUNCTIONAL PDE OPERATORS IN A DATA-DRIVEN SETTING

ANNA DODSON, ARVID LEVANDER, EVAN OBERSTEIN, & ELLA PALACIOS

Applied Mathematics, University of Washington, Seattle, WA

Abstract. In this work we investigate, reproduce, and extend the results given in "A Kernel Framework for PDE Discovery and Operator Learning" [3] via methods put forth in both the aforementioned paper and "Discovering governing equations from data by sparse identification of nonlinear dynamical systems" [6]. We consider physical systems arising from the motion of a pendulum and Burger's equation. We train and evaluate on a noisy set of solutions for each physical system and perform cross-validation to obtain the best performing kernels. We then extend the work to a real, unknown dynamical system from NOAA's International Best Track Archive for Climate Stewardship [1, 5, 4] to explore whether we can reconstruct the trajectories in time of tropical cyclones.

1. Introduction and Overview

We live in a world governed by the laws of physics, where physical observations and phenomena can often be described by dynamics in the form of differential equations. Existing heuristic methods require tediously fine-tuning parameters in order to match the given data, and we might be able to employ instead a data-driven approach. Kernel methods are a useful theoretical tool which allow us to discover the relationships between data and its derivatives, thereby enabling us to learn operators which govern the functional relationships within the data. For systems governed by partial differential equations (PDEs), the operator of interest is a partial differential operator. The motivation of this project is to understand and derive equations for PDEs given a set of noisy solution data and then be able to apply this method to a real valued data set.

We investigated three separate data domains, two of which are mathematically defined problems with generated data, the third using data from International Best Track Archive for Climate Stewardship (IBTrACS) data: (i) pendulum data, (ii) Burger's equation data, and (iii) cyclone data.

We begin simply in (i), by using a differential dynamical system for a pendulum that begins at rest with no initial displacement or velocity, given below. For this problem, we define the initial conditions for this system of temporal ordinary differential equations (ODEs) as $u_1(0) = 0 = u_2(0)$.

$$\begin{cases} u_1'(t) &= u_2(t) \\ u_2'(t) &= -k\sin(u_1(t)) + f(t) \end{cases} \tag{1}$$

The force behind the pendulum movement in the system is $f(t)$, a forcing function, drawn independently and identically distributed from a Gaussian process. Our parameter k is given by $\frac{g}{\ell}$, with g being gravity, and ℓ the length of the string between the over-head support and the mass.

The Burgers Equation (ii) is well known for its shock generating properties and is used to model fluid dynamics, nonlinear acoustics, and traffic flow. The canonical form of this commonly parabolic viscous PDE is given as:

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

Date: June 7th, 2024.

We used the viscous Burgers equation with viscosity constant, $\nu = 0.1$, as given in equation 2.

$$u_t = -uu_x + 0.1u_{xx} \tag{2}$$

Normally we expect to see a PDE problem posed also with initial and boundary conditions, but since we already have a numerical solution and are focused on solely recovering the functional form of the PDE, we have no need to consider the initial and boundary conditions.

(iii) Our third investigation poses the question of equation recovery on the IBTrACS dataset, which is the most complete global collection of tropical cyclone data aggregated from multiple agencies to create a unified and publicly available dataset. From this dataset, we used storms from the last three years as a toy dataset from the main dataset since kernel methods do not require the vast amounts of data other methods do. Our treatment of the data in the unsolved problem is described in Section 3.

The algorithmic plan for each of these investigations is to add some small noise to the data, smooth the data, and make an attempt at learning the functional form through Kernel processes. We also repeat the smoothing process on data with 100% noise added to test understand the limits of the RBF and polynomial kernels. In the following section the mathematical theory behind Kernel processes for learning functional forms will be defined and applied.

2. Theoretical Background

We begin with a discussion of the theoretical background behind Operator Learning with a specific emphasis on ODE's and PDE's. We consider: Given a set of pairs $\{u_i, f_i\}_{i=1}^N$ satisfying some differential equation $\mathcal{P}(u_i) = f_i$, can we learn the functional form, for an ODE, $\mathcal{P}(x, \partial^{\alpha_1}u(x), \ldots, d^{\alpha_P}u(x))$, or for a PDE, $\mathcal{P}(x, d^{\alpha_1}u(x), \ldots, d^{\alpha_P}u(x))$? Via Reisz Representer Theorems we have seen that an interpolator of a given dataset $\mathbf{X}$ on a mesh is given as:

$$\mathbf{u}^* = K(\cdot, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y} \tag{3}$$

Where $\mathbf{y}$ is the ground truth labels on the mesh. Then suppose we have a candidate set of functions and want to learn a functional $\mathcal{P}$ given some data and the candidate set to map our learned data to the true data:

$$\overline{\mathcal{P}}\left(x, \partial^{\alpha_1}u(x), \ldots, \partial^{\alpha_P}u(x)\right) = f(x), \quad x \in \Omega$$

In the above case, $\overline{\mathcal{P}}$ serves as a best approximation to the true functional $\mathcal{P}$. A well posed example of this is for the general case of the pendulum problem, as given in equation (1), wherein we must learn two functionals each parameterized by $t, u_1(t), \frac{du_1(t)}{dt}$, and $t, u_2(t), \frac{du_2(t)}{dt}$ respectively. We also wish for the functional to be of the same structure as the original dynamical system, such that:

$$\begin{cases} \overline{\mathcal{P}_1}(t, u_1(t), \frac{du_1(t)}{dt}) & = u_1'(t) \\ \overline{\mathcal{P}_2}(t, u_2(t), \frac{du_2(t)}{dt}) & = u_2'(t). \end{cases} \tag{4}$$

Within the interpolator, we are able to treat $\mathbf{X}$ and $\mathbf{y}$ as a constant with respect to the argument x of $\mathbf{u}^*(x)$, and so, as we are able to identify the true derivatives of $\mathbf{u}^*$ and use this to estimate the derivatives of u, $\mathcal{P}$ and $\overline{\mathcal{P}}$ respectively. Representing the interpolator problem in the terms of derivatives:

$$\partial_{x_k}\mathbf{u}^*(x) = \partial_{x_k}(K(x, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y}) = \partial_{x_k}(K(x, \mathbf{X}))K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y}. \tag{5}$$

In practice, it is straightforward to obtain a relationship between the derivative Kernel, given in equation (5) and the original Kernel, which we'll need to compute $K(\mathbf{X}, \mathbf{X})$. For example, the RBF Kernel, where ℓ is the length scale, is defined as $K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||_2^2}{2\ell^2}\right)$. When we apply the RBF Kernel to the derivative kernels above, by the power rule for derivatives, we have: $\partial_{x_k}(K(x, \mathbf{X}_i)) = -\frac{(x_k - \mathbf{X}_i)}{\ell^2} * K(x, \mathbf{X}_i)$.

We would like to apply this theory to the three problems outlined in Section 1. For each we'd like to learn the functional form on the operators, given some noisy solution data. We begin by learning a smooth function on our PDE data, $u(t, x)$. For generalizing our theory, we'll use $\bar{u}$ in notation from here on out, unless otherwise specified. Our smoothing equation is given by

$$\bar{u}^{(i)}(x) = \mathcal{U}(x, X)\left(\mathcal{U}(X, X) + \lambda_{\mathcal{U}}^2 I\right)^{-1} u^{(i)}(X) \tag{6}$$

with samples $i = 0 \dots I - 1$, where I is the total number of functions that we use to compose our dataset and $\lambda_{\mathcal{U}}$ regularization parameter. We then apply the derivative operator on equation (6), which yields the the set $s(i)_j$, as defined in equation (7).

$$\partial^{\alpha_k} \bar{u}^{(i)}(x_j) = \partial^{\alpha_k} \mathcal{U}(x_j, X)\left(\mathcal{U}(X, X) + \lambda_{\mathcal{U}}^2 I\right)^{-1} u^{(i)}(X)$$

$$s_j^{(i)} = \{x_j, \partial^{\alpha_1} u(x_j), \dots, \partial^{\alpha_P} u(x_j)\}. \tag{7}$$

The concatenation of $s_j^{(i)}$ forms our total data set, S. Then, we use this set, S in our Kernel learning approach, defined in equation (3) to learn the function form of the forcing function, f. We learn the operator $\overline{\mathcal{P}}$,

$$\overline{\mathcal{P}} = K(\cdot, S)\left(K(S, S) + \lambda_{\mathcal{K}}^2 I\right)^{-1} \mathbf{f}, \tag{8}$$

such that $\overline{\mathcal{P}}(S) = f(x)$, $\forall x \in \Omega$. It is worth noting that our sample space Ω follows all properties of a Reproducing Kernel Hilbert Space (RKHS) which allows for the applications of Representer Theorems earlier in this section. Finally, we notate the application of our learned operator on a test data point, s, which gives us the form

$$\overline{\mathcal{P}}(s) = K(s, S)\left(K(S, S) + \lambda_{\mathcal{K}}^2 I\right)^{-1} \mathbf{f}. \tag{9}$$

Several choices remain underdefined in this approach. First, our choice of kernels $\mathcal{U}$ and $\mathcal{K}$ is constrained to the following for the theoretical basis outlined above:

- $\mathcal{U} : \Omega \times \Omega \to \mathbb{R}$ a Mercer kernel with RKHS $\mathcal{H}_{\mathcal{U}}$ in $C^{\mathrm{MP}}(\Omega)$.
- $\mathcal{K} : \mathbb{R}^{JP} \times \mathbb{R}^{JP} \to \mathbb{R}$ with RKHS $\mathcal{H}_K$ in $C_0(\mathbb{R}^{JP})$.

While our choice of parameters $\lambda_{\mathcal{U}}$ and $\lambda_{\mathcal{K}}$ also remain undefined and will be chosen along with other kernel hyperparameters. With this theory in place, we now turn to our experimental setup over the three investigations. In order to apply this theory to problems, we next generate solution data for each problem given in section 1. We will then apply the method on both testing and training data to evaluate the validity as well as potential adjustments needed.

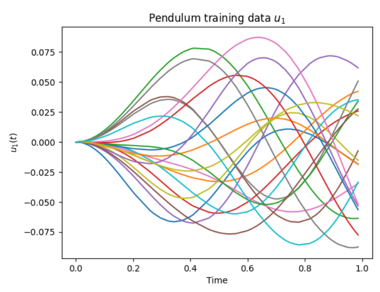
3. Data

Our experiments include data from three domains: position of a pendulum, Burger's equation dynamics, and the paths of tropical cyclones. In order to fully control our experiments in the pendulum and Burger's equation data settings, we hand-generated our data in these settings. To extend our methodology to a real-world setting, we also drew upon the IBTrACS dataset. For our generated datasets, code was written in python using applications of the numpy, scipy, scikit-learn, and SINDy libraries [6].

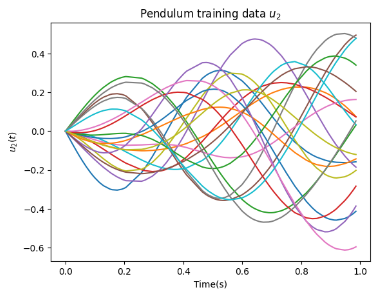
(i) **Pendulum.** The pendulum problem lies within the temporal domain of $t \in [0, 1]$ and, optimally, the spatial range pertains to a realistic value of k, which gives $u(t) \in [-50, 50]$, approximately. We generated data in this domain by solving the dynamical system given in equation (1) via the scipy.integrate library `solve_ivp` function, using the RK45 method, which was used due to it's accuracy and wide range of convergence for solving systems of first order ODE's. We take $I = 20$ of these samples to form our training dataset. In order to form the testing dataset, we perturb $f(t)$ by adding additional, periodic forcing terms, such that the forcing function is now:

$$f_\beta(t) = f(t) + 0.5 \sin(5\pi t).$$

For the test data, we take $I = 50$, and each forcing term $f^i(t) \quad \forall i \in I$ is drawn iid from a Gaussian process. Executing this gives us the training data for each ODE in the system, as presented in Figures 1a and 1b. Then, to generate a noisy training dataset, in order to test our smoothing algorithm we add a small amount of Gaussian noise to the training data. We'll present the noisy dataset further in the results section when we discuss the results of our smoothing algorithm.



(A) Training Data for $u_1(t)$



(B) Training data for $u_2(t)$

(ii) **Viscous Burgers Equation.** The viscous Burgers equation lies in the temporal domain $t \in [0, 10]$ and spatial domain $x \in (-8, 8)$. The range of this problem is $u(x, t) \in [-2, 2]$. The larger temporal domain is due to the shock nature of the viscous Burgers equation and we needed to make sure we had enough data to have a complete picture of the function's behavior.

We utilized the pySINDy package to generate our data. Approaches such as the ones used in [2, 7], require additional JaX implementation of the derivatives which would have increased computational scope of our project considerably. Thankfully, the authors of the SINDy paper included viscous Burger equation data with their experiments. We built upon this, extracting $u(t, x), x,$ and t, and adding some noise to test our smoothing and learning algorithm. We used the pySINDy function, `FiniteDifference_.differentiate()` to calculate u_t from the dataset. This function computes the finite difference scheme derivatives along a given axis in a dataset, we used axis $= 1$ to differentiate along the temporal axis. Doing these steps yields the contour plots of the viscous Burgers equation shown in Figure 2. In order to obtain our test data, we split the data into train and test samples added some sinusoidal noise to the original data to evaluate on a new, noisy scenario.

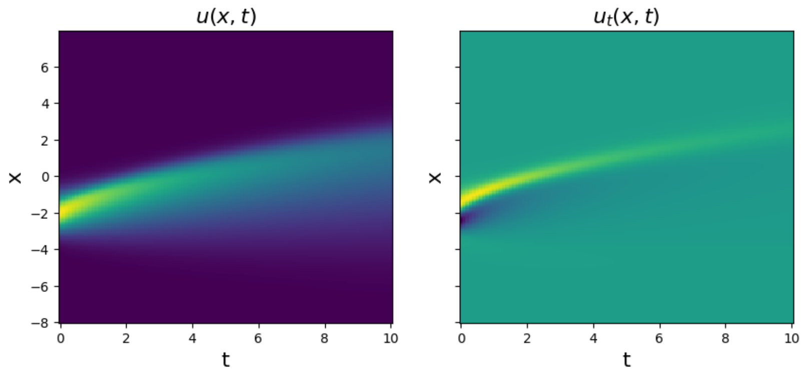


FIGURE 2. $u(x, t)$ and $u_t(x, t)$ contour plotted against x respectively

(iii) **Tropical Cyclones.** The data for the tropical cyclone problem was provided in the IBTrACS dataset. This dataset is the most complete global collection of tropical cyclones available,

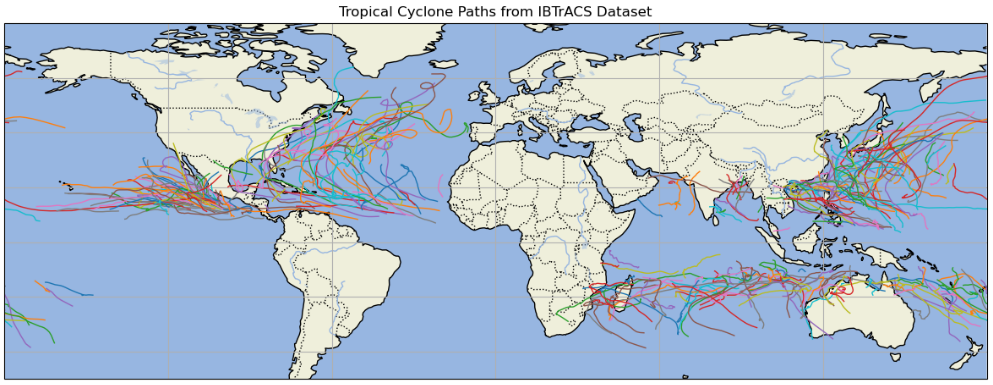


FIGURE 3. The 307 tropical cyclones that form our dataset plotted latitude against longitude on a world map.

combining recent and historical tropical cyclone data from multiple agencies to create a unified, publicly available, best-track dataset. We considered only tropical cyclones that occurred in the past three years, since recent data is both more accurate and spatially comprehensive. On these data, we performed several rounds of filtering: First, we removed data that was not temporally rich enough (fewer than ten data points). We retrieved the relative time and then performed cubic spline interpolation on the remaining 307 trajectories to fill NaN values and ensure all data was on the same time grid of .025 hour increments. Of this data, we truncated all trajectories to contain 80 time points on the scale $[0, 9.875]$ hours.

From all of the data fields in the set, we consider just the lat/long and time of the trajectories and obtain the derivatives via numerical differentiation. That raw data has been projected onto a two-dimensional world map and shown in Figure 3. While we needed to add noise to the pendulum and viscous burgers data, we do not need to do this for the tropical cyclone data since it's "real-world" data and therefore noisiness is a consequence of our measurements containing some observation error.

4. Experimental Setup

4.1. **Algorithmic Design and Implementation.** Our experimental approach crosses several dimensions. We consider 3 domains, 2 kernel choices with 2-3 tunable hyperparameters each, and several noise scenarios. Our experimentation follows a simple recipe: after data generation, we smooth the data for each problem using some kernel $\mathcal{U}$ in the set {RBF, Polynomial}, normalize it, then apply the kernel-based operator learning approach using a kernel $\mathcal{K}$ in {RBF, Polynomial}. The nuances of the algorithm design are revealed in choice of $\mathcal{U}$ and $\mathcal{K}$, as well as hyper-parameter tuning: $\ell, \lambda_{\mathcal{K}}$ for RBF kernel as outlined in Equation (2) and λ, c, d for the Polynomial kernel defined as $K(x, y) = (c + x^T y)^d$. The code for all of our algorithms can be found here.

(i) **Pendulum.** For the pendulum, we were able to implement the smoothing algorithm described in equations (5) and (6). For our set $s_j^{(i)}$ as defined in (7), we chose the derivative space to be $\{\bar{u}_1^{(i)}(t_j), \bar{u}_2^{(i)}(t_j), \bar{u}_1'^{(i)}(t_j), \bar{u}_2'^{(i)}(t_j)\}$. In order to identify optimal parameters, for both RBF and polynomial kernels in smoothing, we used subsets of the training data in a 5-fold cross-validation to find the best-fitting kernel for the test data. We grid-searched the hyperparameter space $\ell, \lambda_{\mathcal{K}}$ for the RBF kernel in a 50x50 2 dimensional grid, and λ, c, d for the Polynomial kernel, with a 50x50 2 dimensional grid for each choice of $d = 2 \ldots 5$. We then evaluated the best-performing kernel(s) on the test dataset, which includes the additional noise term, as described in 3 (i). We visually compare the data for the operator $\overline{\mathcal{P}}_1(S)$ and $\overline{\mathcal{P}}_2(S)$ trajectories compared to the true solution for each sample $s_i \in S$ for $\overline{\mathcal{P}}_1$, the true solution is given by $u_2^{(i)}(t)$, for $\overline{\mathcal{P}}_2$, the true solution is $-\frac{g}{l}sin(u_1^{(i)}(t)) + f^{(i)})$.

(ii) **Viscous Burgers Equation.** Having only one surface for the Burger's equation we treated each $u(x_i, t)$ as a sample. Then we took the same kernel approach but only considering the RBF Kernel. The smoothing hyper-parameters were identified from a mix grid-search and hand tuning. Hyper-parameters for PDE-discovery was found using 3-fold cross validation on the training data with MSE as loss. For the context of this problem, the set $s_j^{(i)}$ was considered to be $\{\bar{u}^{(i)}(x_j, t_j), \partial_x \bar{u}^{(i)}(x_j, t_j), \partial_{xx}\bar{u}^{(i)}(x_j, t_j)\}$. We then evaluated the method on test data which was prepared by perturbing the given dataset with a sinusoidal function. We compare our results for $\overline{\mathcal{P}}(s)$ with the true solution u_t both visually and by using MSE.

(iii) **Tropical Cyclones.** For each of the previous sections the data we were considering was generated on the same meshgrid and had a known ground truth. We had no such luxury on the cyclone trajectory data. Our data preparation approach is as outlined above. Then, due to the unique nature and path of each storm, as well as unique total lengths, we applied the smoothing step to each storm separately. Since we did not have a ground truth, we applied a heuristic visual sanity test to try out different parameters. Luckily, we were easily able to distinguish poor and good results as determined by the choice of ℓ and $\lambda_{\mathcal{U}}$. Another major drawback of the open domain data is that we did not have a forcing term included in the dataset, nor do we know the true nature of the PDE. Thus, the power of the kernel approach is slightly undermined. For this problem, equation (7) looked like $s_j^{(i)} = \{t_j, \bar{u}_1^{(i)}(t_j), \bar{u}_2^{(i)}(t_j), \bar{u}_1'^{(i)}(t_j), \bar{u}_2'^{(i)}(t_j), \bar{u}_1''^{(i)}(t_j), \bar{u}_2''^{(i)}(t_j)\}$, where $\bar{u}_1^{(i)}$ and $\bar{u}_2^{(i)}$ are respectively the smoothed longitude and latitude of the i^{th} cyclone. Similar to the pendulum approach, we will attempt to recover the time derivative in each of the lat/long directions.

5. Computational Results

5.1. Pendulum.

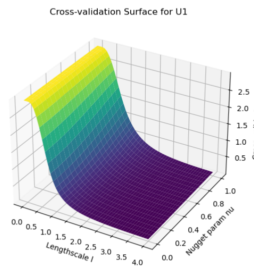
Following the approach described in the previous section on both parts of the ODE, we are able to recover the functional form of both u_1 and u_2 for the pendulum data. For smoothing the data we found that an optimal lengthscale for the RBF Kernel for this problem was $\ell = 0.01$ and for a fifth-order ($d = 5$) Polynomial Kernel, optimal free-parameter, $c = 15$. We present the hyper-parameters and corresponding MSE for learning, K the data[1] in Table 5. We can see from the data in Table 5 that the RBF Kernel for both smoothing and learning performed the best for this data. We present a plot of the noisy training data, the smoothed data and the learned functional in Figure 6.

The primary difficulties we encountered in the pendulum system was correctly normalizing the data to obtain an ℓ within the domain of the problem. We found that if we normalized the data and then applied the Kernel method for learning, the nature of the derivatives in the problem would result in the length scale being well outside of the domain tolerance. The fix for this was to apply the normalization on the data after we'd done the derivatives. We also found that our choice of scalar was resulting in problematic ℓ values. Rather than using the standard-scaler from sklearn, which takes the standard statistical z-score, we found that the min-max scaler performed much better. The min-max scaler scales and translates each feature individually such that it is in the given range on the training set i.e. between 0 and 1. This gave us better MSE and much more reasonable ℓ.

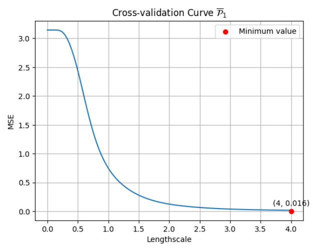
5.2. Burgers Equation Results.

Given the nice performance of the RBF Kernel on the Pendulum data, we also applied RBF smoothing and learning to the viscous burgers equation. We found an optimal length-scale $\ell = 9.83$, which resulted in a cross-validation training MSE of 0.003 and MSE of 0.03 on the testing data. For generating the test data we perturb the training data by

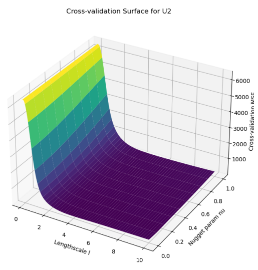
[1]For each Kernel learning method in this table, the same kernel method for smoothing was used.



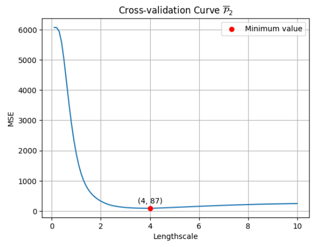
(A) CV surface for $\overline{P}_1$



(B) $\overline{P}_1$ curve, $\lambda_{\mathcal{K}} : 10^{-2}$



(C) CV surface for $\overline{P}_2$

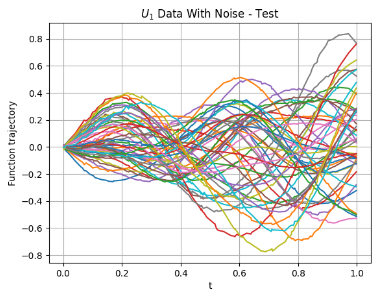


(D) $\overline{P}_2$ curve, $\lambda_{\mathcal{K}} : 1e^{-8}$

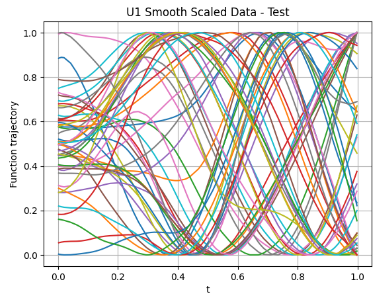
Optimal Hyperparameters			
(i): $\mathcal{U}$	u_1	RBF	$\ell = 0.1,\ \lambda_{\mathcal{U}} = 1.0e^{-3}$
		Poly	$d = 5,\ c = 15,\ \lambda_{\mathcal{U}} = 1.0e^{-3}$
	u_2	RBF	$\ell = 0.1,\ \lambda_{\mathcal{U}} = 1.0e^{-3}$
		Poly	$d = 5,\ c = 15,\ \lambda_{\mathcal{U}} = 1.0e^{-3}$
(ii): $\mathcal{K}$	$\mathcal{P}_1$	RBF	$\ell = 4.0,\ \lambda_{\mathcal{K}} = 1e^{-8}$
		Poly	$d = 3,\ c = 0.716,\ \lambda_{\mathcal{K}} = 1e^{-3}$
	$\mathcal{P}_2$	RBF	$\ell = 4.87,\ \lambda_{\mathcal{K}} = 1e^{-8}$
		Poly	$d = 3,\ c = 1.0,\ \lambda_{\mathcal{K}} = 1e^{-3}$

FIGURE 4. (above) Optimal Hyper-parameters and corresponding minimum squared error for the pendulum problem for each Kernel Method used in learning.

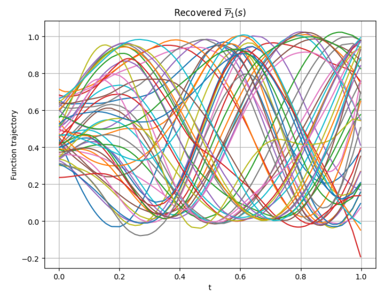
FIGURE 5. (left) Example of Cross-validation: RBF Kernel $\mathcal{K}$, Pendulum Data.



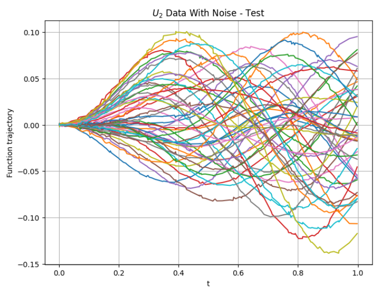
(A) Noisy data for u_1



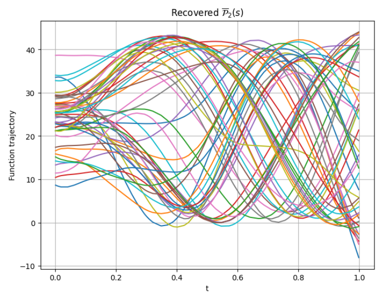
(B) u_1 smoothed with RBF.



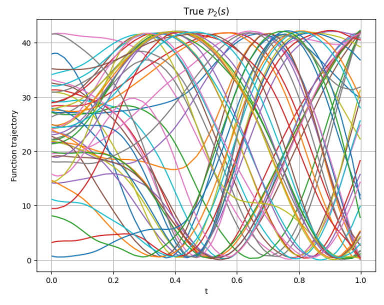
(C) Recovered $\overline{\mathcal{P}}_1$



(D) Noisy data for u_2



(E) Recovered $\overline{\mathcal{P}}_2$



(F) Full recovered solution

FIGURE 6. All data, noisy through smoothed, for RBF smoothing and learning recovery, and recovered functional, applied to S, the test set.

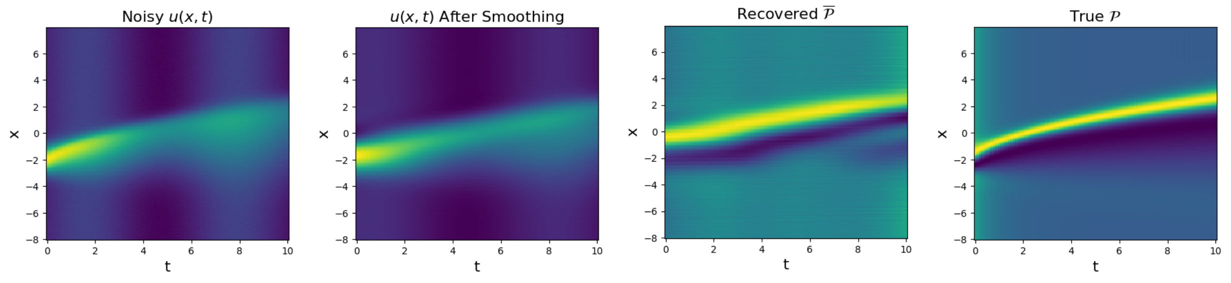
adding $0.1 \sin(t)$, which explains the difference in MSE for testing versus training. Given the domain and range of this dataset these are reasonable values and further emphasize the performance of the RBF Kernel on datasets, regardless of the structure. We present heat-maps of the Burgers

Model	RBF Kernel	Polynomial Kernel
$\overline{\mathcal{P}}_1(s)$	0.0018	0.0094
$\overline{\mathcal{P}}_2(s)$	9.23	45.83

Table 1. Mean Squared Error for $\overline{\mathcal{P}}_1(s)$ and $\overline{\mathcal{P}}_2(s)$ with RBF and Polynomial Kernels for both steps

dataset for the noisy data, smoothed data, recovered functional the the true values in Figure 7.

We faced challenges with Gibbs phenomenon when smoothing $u(t,x)$. The Gibbs phenomenon would create small surface waves on $u(t,x)$ which were barely noticeable on $u(t,x)$ but would create large errors for u_t, u_x, and u_{xx} and consequently errors in the PDE discovery step. Hand-tuning was therefore required to get a flatter surface and more accurate partial derivatives. We faced the same issues with normalization as we did with the pendulum so we applied a min-max scaler rather than a traditional z-score and recovered more reasonable length scales within our problem domain. An additional challenge we encountered was with the shock nature of the problem: the viscous burgers equation can generate a shock which would be a non-differentiable point. We needed to make sure that our Kernel method both accounted for this shock point in the data without eliminating it all together in smoothing. We theorize that a smoothing Kernel, such as a low degree polynomial Kernel would remove the shock nature from our problem. We were unable to test this due to time constraints on the project.



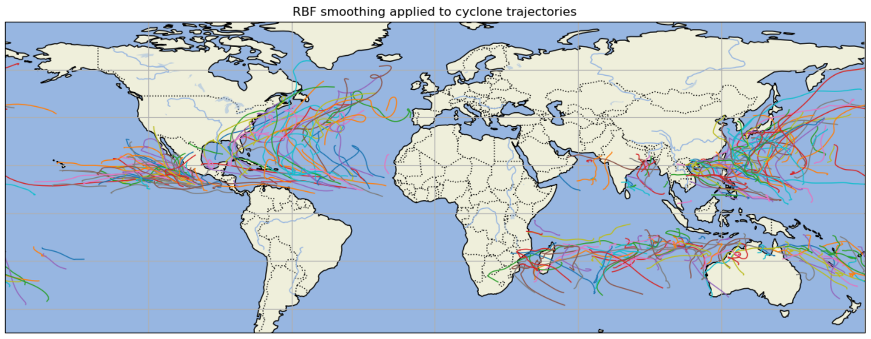
(A) Noisy data of $u(x,t)$ (B) Smoothed data (C) Recovered $\overline{\mathcal{P}}$ (D) True $\mathcal{P}$

Figure 7. Results for the viscous Burgers Equation with the RBF Kernel applied

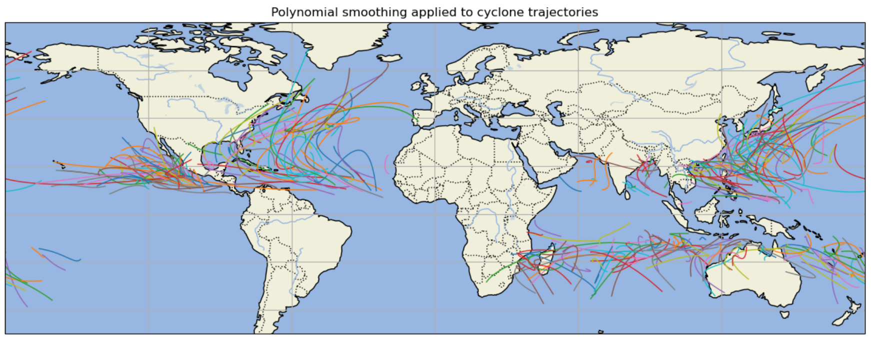
5.3. **Tropical Cyclone Data Results.** Cyclone data was more difficult to obtain a meaningful result over due to the lack of a forcing term included in our data. However, the smoothing properties of the kernel are still very useful in de-noising the trajectory to learn a potentially low order representation of the PDE. In this section, we tested out a few ideas and report on our results.

1. Smoothing the data. In smoothing our data, we tested the polynomial and RBF kernels. We were able to tune the hyperparameters to recover smooth parabolic curves which follow the original trajectory of the data very well. In order to illustrate the effectiveness, we present the original trajectories in figure 3 as well as the smoothed trajectories in the tuned RBF and polynomial kernel cases in figure 9.

2. Learning the spatial operator. We pose the problem of trying to recover the trajectory given the spatial lat/long and its derivatives. At this point, we discovered our dataset is quite ill-conditioned. Some of the time derivatives were extremely large even when computed to second-order precision, smoothed, and gradient clipped. As a result, even with a 3-fold cross-validation approach we observe very poor results after numerically integrating the time derivatives dt/d_{lat} and dt/d_{long} we recover via the operator learning.

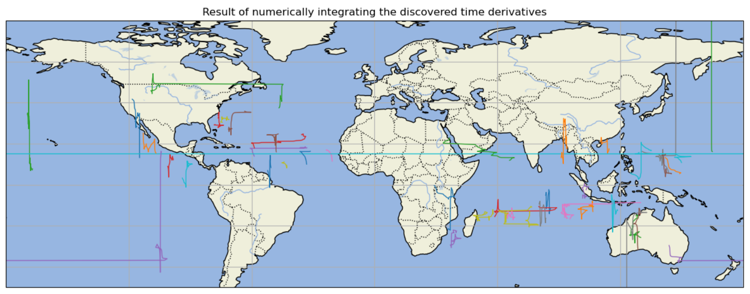


(A) RBF smoothing applied to cyclone trajectories

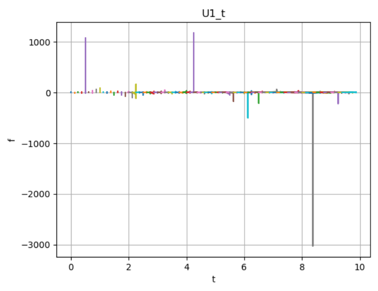


(B) Polynomial smoothing of cyclone trajectories

FIGURE 8. RBF vs Polynomial smoothing on cyclone trajectory data


Result of numerically integrating the discovered time derivatives

(A) Numerical integration of learned time derivative operator for cyclone trajectories



(B) The ill-conditioning issue with time derivatives

FIGURE 9. Numerical Instability on real data

As shown in Figure 9, our approach fails. Even though we are able to smooth our data sufficiently, this does not translate to the stability of the method since the derivatives with respect to the trajectory can be very small (or large).

6. Summary and Conclusions

In this project, we combine physical knowledge, mathematical theory, and observations to infer the structure of highly complex systems. We accomplish this through the use of kernel methods to learn the form of a functional over a (possibly noisy) dataset. We show the physical prowess of such a technique when applied to a variety of domains, from a simple pendulum to the viscous Burger's equation to the trajectory of cyclones in a real world setting. During our experimentation, we explored the trade-offs between various parameterizations of the same two-step method to learn an operator. Along the way, we experimented broadly, including: choice of kernel, grid search cross-validation of hyperparameters, performance and resilience to noisy data, normalization, exploring different differentiation techniques and packages, and even modifying the functional basis for each problem to obtain our best results. In this report, we mainly focus on the former three, though the latter three were also tinkered with along the way so as to achieve the best final outcome.

The best-performing model in the Pendulum problem was a two-step approach with Mercer kernels $\mathcal{U}$ and $\mathcal{K}$ which learned the functionals $\overline{\mathcal{P}}_1(\{\bar{u}_1^{(i)}(t_j), \bar{u}_2^{(i)}(t_j), \bar{u}_1'^{(i)}(t_j), \bar{u}_2'^{(i)}(t_j)\})$ and $\overline{\mathcal{P}}_2(\{\bar{u}_1^{(i)}(t_j), \bar{u}_2^{(i)}(t_j), \bar{u}_1'^{(i)}(t_j), \bar{u}_2'^{(i)}(t_j)\})$. We found the model that performed the best for $\overline{\mathcal{P}}_1$ was the RBF kernel for $\mathcal{U}$ with $\ell = .1, \lambda_{\mathcal{U}} = 1.0e^{-3)}$ and RBF kernel for $\mathcal{K}$ with $\ell = 4.0, \lambda_{\mathcal{U}} = 1.0e^{-8)}$, which had an MSE over 50 test pendulums of .0018. For $\overline{\mathcal{P}}_2$ the most performant configuration was the RBF kernel for $\mathcal{U}$ with $\ell = .1, \lambda_{\mathcal{U}} = 1.0e^{-3)}$ and RBF kernel for $\mathcal{K}$ with $\ell = 4.87, \lambda_{\mathcal{U}} = 1.0e^{-8)}$ which had an MSE over 50 test pendulums of 9.23. On the viscous Burger data, we found an optimal length-scale $\ell = 9.83$, which resulted in a cross-validation training MSE of 0.003 and MSE of 0.03 on the testing data. Finally, in a novel domain which consisted of real-world tropical cyclones data, we were able to achieve excellent smoothing results with $\mathcal{U}$ being an RBF kernel with $\ell = 1$ and $\lambda_{\mathcal{U}} = 3$ and determined the dynamics of the system under the assumption that $\overline{\mathcal{P}}$ was a function of $lat, long, lat', long', lat'', long''$ and had the functional form u_t in both the lat and long direction. However, we were unable to recover the function in this form due to numerical instability.

Several next areas of exploration and research stood out to us. Firstly, we could explore further each technique's robustness to noise. We conducted a few experiments in the pendulum data with 100% noise and were able to still see a decent trend recovered in the smoothing step, but we didn't repeat this for the operator recovery step. Additionally, working on real-world data requires a bit more engineering design, digging deeper into the tropical cyclone data to determine whether some other variables, like wind or precipitation, might be functions to try and learn - not just the position of the cyclone. Perhaps the time derivative would be better conditioned if we did this approach.

In conclusion, we have thoroughly tested the robustness of kernel methods in discovering PDE operators. In comparison to other methods such as SINDy which require a library of combinations, the kernel approach can be useful in cases where the feature set is very unknown, since the set of possible features can easy be expanded in the candidate function basis.

Acknowledgements

Thank you to each of the members of this group for the meaningful contributions to this project, and our deep gratitude also to Bamdad Hosseini for an excellent course and his guidance during the construction of this project.

References

[1] Noaa's international best track archive for climate stewardship (ibtracs) data, 2024.

[2] O. H. S. A. M. Chen Yifen, Hosseini Bamdad. Solving and learning nonlinear pdes with gaussian processes. *Journal of Computational Physics*, 2021.

[3] S. Z. Da Long, Nicole Mrvaljević and B. Hosseini. A kernel framework for pde discovery and operator learning. Submitted by Authors in 2022.

[4] K. K. S. Knapp, Diamond. The international best track archive for climate stewardship (ibtracs) project, version 4. 2018.

[5] L. D. N. Knapp, Kruk. The international best track archive for climate stewardship (ibtracs): Unifying tropical cyclone best track data. *Bulletin of the American Meteoroglical Society*, 91:363–376, 2010.

[6] J. L. P. Steven L. Brunton, J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932 – 3937, 2016.

[7] M. Zingale. *Tutorial on Computational Astrophysics.* 2021. https://zingale.github.io/comp_astro_tutorial/advection_euler/burgers/burgers-methods.html.