LEARNING REDUCED DYNAMICS FROM FULL EQUATIONS

CADE BALLEW, ELLIE BYRNES, ALEX HSU, SARA ICHINAGA, AND KAITLYNN LILLY

Abstract. We investigate the feasibility of Sparse Identification of Nonlinear Dynamics (SINDy) [3] as a model reduction technique for the full Euler water wave problem. We apply SINDy to data simulated from the full Euler water wave problem in the regime in which it is well-approximated by the Korteweg–de Vries (KdV) equation for long waves in shallow water. We find that SINDy cannot recover the KdV equation in this case, even after performing cross-validation; however, we determine that SINDy *can* recover the KdV equation given data simulated directly from the KdV equation itself. Furthermore, we find that given cnoidal solutions to the full water wave problem in the KdV regime, SINDy can recover the KdV equation after cross-validation, but not consistently. We conclude that in its current state, SINDy cannot reliably be used as a model reduction technique.

1. Introduction

In recent years, a variety of techniques have emerged for inferring the dynamics from snapshots of trajectories. One such paradigm is through Sparse Identification of Nonlinear Dynamics (SINDy), a relatively novel yet extremely applicable technique that utilizes sparse regression to extract governing equations from data [3]. If we consider a dynamical system of the form

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)), \tag{1}$$

and are given a time history of the state $\mathbf{x}(t)$ and either a history of or approximations to $\dot{\mathbf{x}}(t)$, SINDy directly approximates f through sparse regression on a fixed feature library.

Rather than using it for system identification, we consider SINDy from the perspective of reduced-order modeling: given data from a larger and more complicated system, can we utlize SINDy to obtain a simpler model that encapsulates our data? More specifically, as

Date: June 7, 2023.

our larger system, we consider the one-dimensional Euler water wave equations:

$$\begin{cases} \phi_{zz} + \epsilon\phi_{xx} = 0, & \text{for } -1 < z < \epsilon\zeta, \\ \phi_z = 0, & \text{at} \quad z = -1, \\ \epsilon\zeta_t + \epsilon^2\phi_x\zeta_x = \phi_z, & \text{at} \quad z = \epsilon\zeta, \\ \phi_t + \zeta + \dfrac{1}{2}\phi_z^2 + \dfrac{\epsilon}{2}\phi_x^2 = 0, & \text{at} \quad z = \epsilon\zeta. \end{cases}$$

In the asymptotic regime of long waves in shallow water, the dynamics of the surface of the Euler water waves can be described by the Korteweg–de Vries (KdV) equation[1]:

$$u_t + 6uu_x + u_{xxx} = 0.$$

The analysis for this reduction is highly nontrivial and requires a number of seemingly arbitrary intermediary choices. The variable u in the KdV equation corresponds to the height of the water in the Euler equations which acts as a time dependent boundary and is difficult to extract from the Euler equations directly.

In general, governing equations for problems in fluids tend to be very complicated, making them both difficult to analyze and to solve numerically or analytically. As such, we often require these sorts of asymptotic reductions to say anything about the solutions. We explore the idea of using SINDy to automatically extract candidate reductions of water wave equations. More specifically, we apply SINDy to data corresponding to the Euler water wave problem in the asymptotic regimes in which the KdV equation is thought to be a valid approximation.

In section 2, we re-examine and reproduce the SINDy analysis of the Lorenz system found in [3]. In section 3, we apply SINDy to the Euler water wave problem and investigate under what conditions the KdV equation can be recovered, if at all. We summarize our results in section 4 and relegate proofs of properties of the Euler water wave equations and the KdV equation to the appendices. Code used to generate the plots in this paper can be found at [2].

[1]In Appendix B, we show how the KdV equation can be derived from the Euler water wave equations.

2. Reproduction of results in paper

2.1. **Background.** Given snapshot measurements $\mathbf{x}(t) \in \mathbb{R}^n$ observed at times $\{t_j\}_{j=1}^m$, the SINDy algorithm assumes dynamics of the form (1) and seeks to approximate the governing dynamics by

$$f(\mathbf{x}) \approx \sum_{i=1}^{\ell} \theta_i(\mathbf{x})\xi_i,$$

i.e., as a sparse linear combination of candidate basis functions $\{\theta_i\}_{i=1}^{\ell}$ evaluated on the state variables $\mathbf{x}$ where the majority of the coefficients ξ_i are zero.

In order to solve for f, we arrange our state variables $\mathbf{x}(t)$ and their associated time derivatives $\dot{\mathbf{x}}(t)$ into the columns of the matrices $\mathbf{X}, \dot{\mathbf{X}} \in \mathbb{R}^{m\times n}$, respectively. These are given by

$$\mathbf{X} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix}, \qquad \dot{\mathbf{X}} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \dots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \dots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \dots & \dot{x}_n(t_m) \end{bmatrix}.$$

Note that if our system is of kth order, then we must include the first $k-1$ derivatives in the state variables $\mathbf{X}$. If the derivatives $\dot{\mathbf{x}}$ are not observed directly, they may be approximated from $\mathbf{x}$ via some numerical differentiation scheme such as finite difference or spectral differentiation.

The feature matrix $\boldsymbol{\Theta}(\mathbf{X}) \in \mathbb{R}^{m\times \ell}$ is defined as

$$\boldsymbol{\Theta}(\mathbf{X}) = \begin{bmatrix} | & | & & | \\ \theta_1(\mathbf{x}) & \theta_2(\mathbf{x}) & \dots & \theta_\ell(\mathbf{x}) \\ | & | & & | \end{bmatrix},$$

so that the columns of $\boldsymbol{\Theta}(\mathbf{X})$ contain our candidate basis functions evaluated at the state variables. Then,

$$\dot{\mathbf{X}} \approx \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}, \tag{2}$$

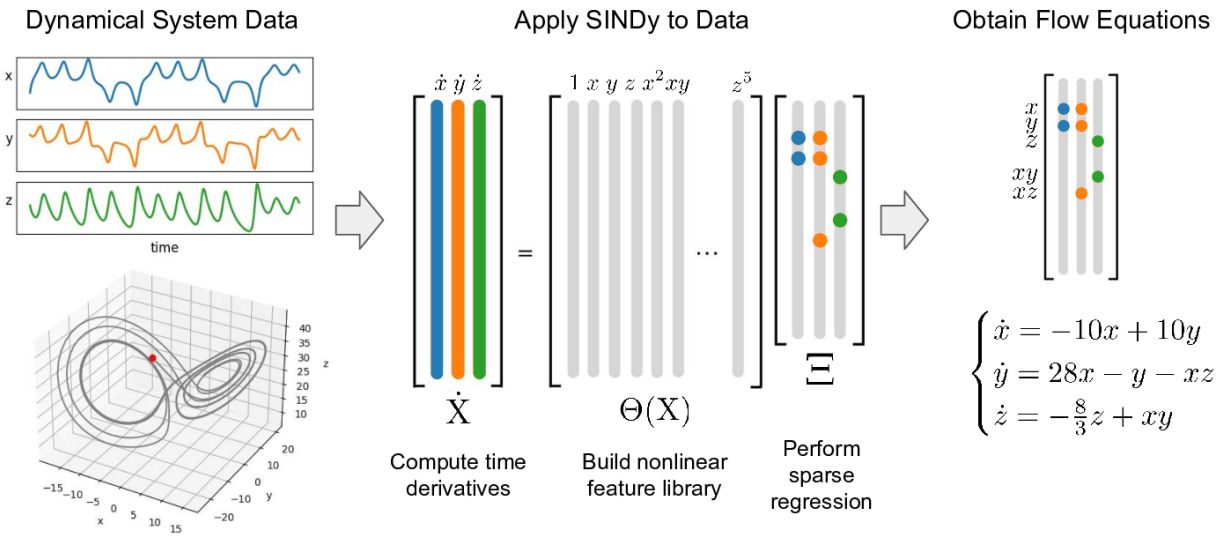

FIGURE 1. Outline of the SINDy algorithm, as proposed in [3]. First, data is collected from a dynamical system of interest. Derivatives are then computed in the second step, along with a feature library evaluated on the observed state variables. Finally, sparse regression is performed to recover governing equations.

where we solve for the coefficient matrix $\boldsymbol{\Xi} \in \mathbb{R}^{\ell \times n}$ via a sparse regression technique such as sequentially thresholded least-squares. This entire process is summarized in Figure 1.

2.2. **Examples.** We begin by re-examining the performance of the SINDy approach when applied to data derived from canonical nonlinear dynamical systems. In particular, we consider the Lorenz system, which is defined by the system of ordinary differential equations

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(\rho - z) - y, \\ \dot{z} = xy - \beta z. \end{cases}$$

This system defines an attractor in three-dimensional space and exhibits chaos for particular choices of the real-valued parameters σ, ρ, and β. In accordance with the original SINDy analysis, we examine this system parameterized with $(\sigma, \rho, \beta) = (10, 28, 8/3)$ and with initial condition $(x_0, y_0, z_0) = (-8, 8, 27)$ [3].

Using the PySINDy Python package [4,8], we apply SINDy to Lorenz system data collected at $m = 5000$ temporal snapshots uniformly sampled in time with $\Delta t = 0.002$. See Figure 1

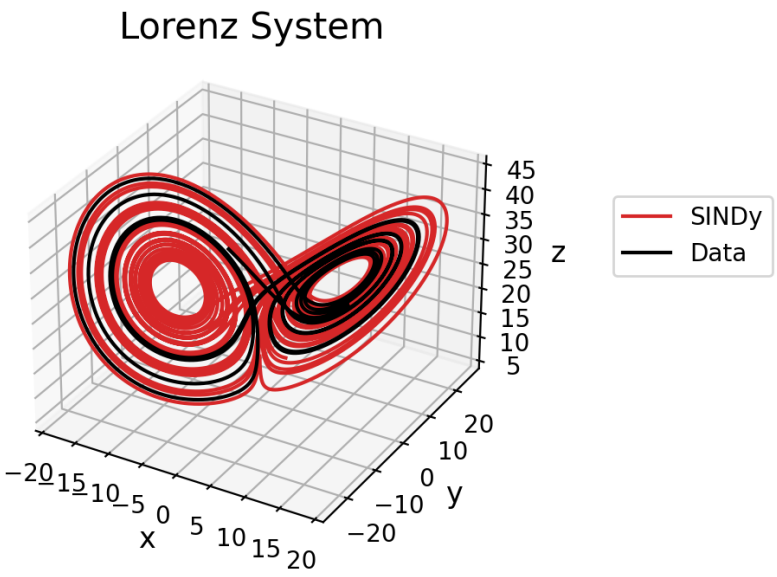

FIGURE 2. Simulated Lorenz system data (red) predicted using a SINDy model trained on a short noise-free Lorenz system trajectory (black).

for an example of such time-series data. To do so, we populate the rows of $\mathbf{X}$ with temporal snapshots of the state $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ and compute the entries of $\dot{\mathbf{X}}$ from $\mathbf{x}$ via second-order centered finite difference. We construct $\boldsymbol{\Theta}(\mathbf{X})$ such that it contains polynomials of $\mathbf{x}$ up to order five. Finally, we perform sequentially thresholded ridge regression in order to solve (2) and obtain the following dynamics model:

$$\begin{cases} \dot{x} = 9.999(y - x), \\ \dot{y} = x(27.992 - z) - 0.999y, \\ \dot{z} = xy - 2.666z. \end{cases}$$

Notice that we accurately recover the true Lorenz system equations within 0.3% of the true coefficient values, as shown in the original SINDy analysis of the Lorenz system [3]. Furthermore, we can use PySINDy to simulate our learned Lorenz model forward in time, the results of which are visualized in Figure 2. From this, we see that in addition to being able to recover governing equations, we can use SINDy to interpolate as well as extrapolate state observations.

3. Novel results

3.1. **Preliminary results.** In this section, we utilize data numerically generated using the methodologies of Appendix A. Following the derivation from the Euler water wave equations to KdV in Appendix B, we expect the water wave dynamics of the profile of a traveling wave to governed by the following form of KdV:

$$2cu' + 3uu' + \frac{1}{3}u''' = 0,$$

where u is the traveling wave coordinate with independent variable $x - ct$, and c is the wave speed. In our implementation of SINDy, the governing equations are outputted as a system of first order equations. As such, we use reduction of order to reduce our expected form of KdV into a system of 3 first order equations. Doing this results in the following set of expected governing equations:

$$\begin{cases} \dot{u} &= v, \\ \dot{v} &= w, \\ \dot{w} &= -6cv - 9uv. \end{cases}$$

We first implement SINDy on numerically simulated water wave data in the regime where KdV well approximates the dynamics. We initially simulate one period of the profiles of periodic water waves traveling at a fixed speed. Running this data in SINDy without hyper-tuning any parameters results in the following set of governing equations:

$$\begin{cases} \dot{u} &= v, \\ \dot{v} &= w, \\ \dot{w} &= 1021.713v - 3612480.532uv + 662.365vw. \end{cases}$$

Clearly, we get incorrect coefficients and an additional vw term. That being said, we do recover a greatly reduced system when compared to the Euler water wave equations from which the data was simulated, demonstrating the potential of SINDy to be used as a model reduction technique.

Fortunately, by using an appropriate scaling transformation on all variables (both independent and dependent) of the KdV equation, one can choose any coefficients of the equation that they please. As such, even though SINDy outputs the "incorrect" coefficients, we can always use a scaling transformation to choose any coefficients to satisfy the KdV equation. A proof of this is given in Appendix C.

Unfortunately, there is still an issue with the extra vw term that SINDy outputs. As such, we see that SINDy is not returning just the KdV equation as we might expect. While SINDy does not output the KdV equation, it is still possible that what SINDy *does* output could serve as a valid approximation to the full Euler water wave equations. To test this, we numerically compute the solution of the dynamics SINDy produces and compare them with those of the water wave data that SINDy is trained on. A plot of these is shown in Figure 3. From this, we can see that while perhaps one period of the governing equations SINDy outputs could well approximate the water wave equations up to a horizontal shift, there are several periods of nonzero data not present in the true data. As such, we cannot expect the governing equations that SINDy outputs to further our understanding of water waves. This demonstrates that SINDy is failing to serve as a good model reduction technique with our current method and data.

3.2. **Investigation of initial failures.** Now, we investigate why our preliminary results are failing to serve as a successful model reduction technique. One possibility is that SINDy is not optimally performing sparse regression. In this case, we can use cross-validation on several different numerically simulated data sets in order to hypertune the parameters used when implementing SINDy on our test data. Doing this should ensure that SINDy is performing optimally on any given data set.

Another possibility is that we are not properly using SINDy to find KdV from KdV-like data. To determine if this is the case, we can provide SINDy with simulated KdV data rather than simulated water wave data. If SINDy can properly determine KdV as the governing equation given proper KdV data, then we know that the numerically simulated water wave data could potentially be improved to better represent KdV and thus allow SINDy to more

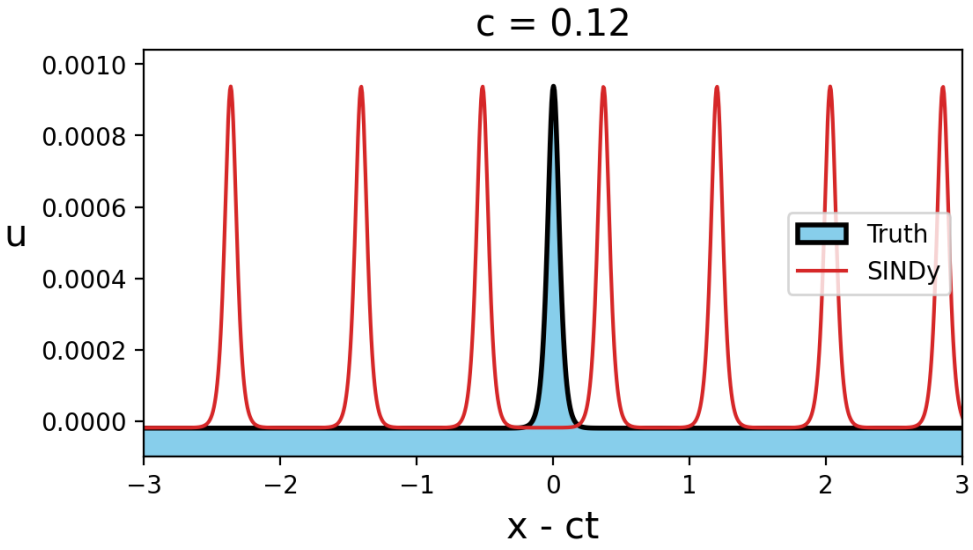

FIGURE 3. An initial SINDy estimate of governing equations to water waves simulated in the KdV regime (red) with a true KdV approximation to data (black).

accurately determine KdV as the governing equations of the dynamics. Alternatively, if SINDy cannot determine KdV as the governing equations given directly simulated KdV data, then we know that we have found a limitation of SINDy and must take alternative measures to determine if SINDy can be used as a proper model reduction technique. However, there are two options for which data to use: fully time-dependent KdV data and the profiles of its traveling wave solutions. Since the former is far more general, we try this first.

We first numerically solve the KdV equation using an exponential integrator. Our solution is shown in Figure 4, and we find that with this data, SINDy is able to reproduce the KdV equations after some parameter tuning. Not only is this another reproduction of the results of the original SINDy paper, but it is also a confirmation that we were correctly utilizing SINDy to identify KdV. However, there is a significant difference in that this data is fully time-dependent data whereas our original data is a stationary solution of the equations in a moving frame. Thus, we should also try to verify that SINDy can identify KdV from its traveling solutions.

KdV has two well-known traveling solutions. One is the one-soliton solution given by

$$u(x,t) = \frac{4}{3}a^2 \operatorname{sech}^2(a(x - (2a^2t/3))),$$

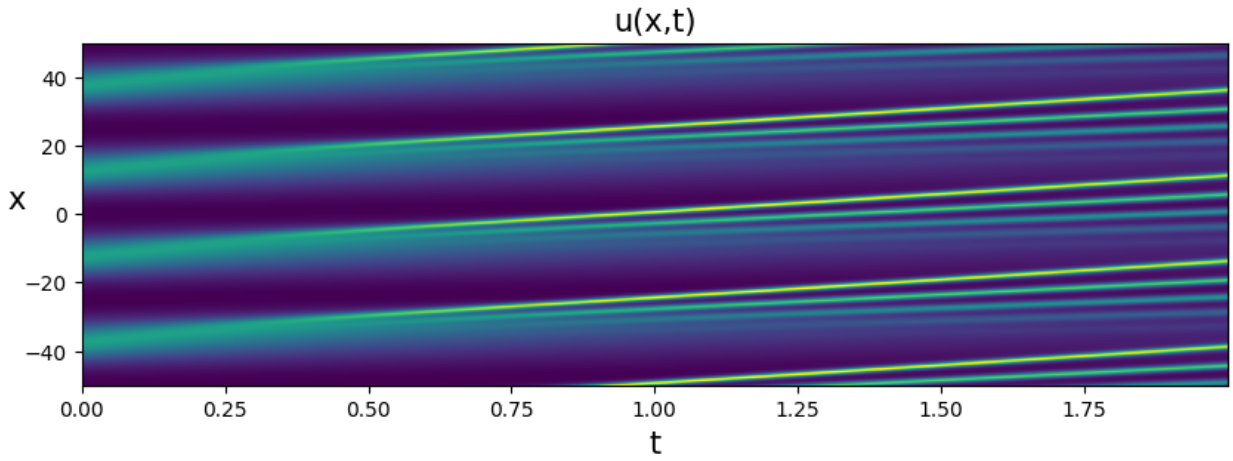

FIGURE 4. A plot of the solution of the time-dependent KdV equation for a general initial condition. Plugging this data into SINDy gives the equation $u_t = -1.037u_{xxx} - 6.090uu_x$, and a numerical solution of the found equations with the same initial condition reproduces this data.

where a is an amplitude parameter. The other is known as the cnoidal solution and is represented using the Jacobi elliptic cosine function, cn. This solution is given by

$$u(x,t) = b + 2k^2\gamma^2 \mathrm{cn}^2(\gamma(x - (6b + (4(2k^2 - 1)\gamma)))t, k^2),$$

where b is the minimum value of the solution and γ is a width parameter. The elliptic modulus k is a bit trickier to characterize, but it corresponds closely to the non-linearity of the solution, with the solutions being a sine wave for $k = 0$, and a soliton solution for $k = 1$. We use these true solutions to KdV to generate traveling KdV data. This data is then fed into SINDy to determine if the numerically simulated water wave data is the primary issue in getting SINDy to work as a proper model reduction technique or if it is a difficulty of SINDy itself.

Unfortunately, SINDy fails to work consistently for these problems, only finding the original equations and fitting the data for a few sets of waves, even after optimally tuning parameters. This behavior is shown in Figure 5. What's interesting here is which of those waves SINDy is accurate on; it finds the KdV equation for tall, narrow waves, but not for long waves in shallow water. These are waves that are in a way unique to KdV in that they are solutions to the equations, but they are not from the regime the equation is derived in.

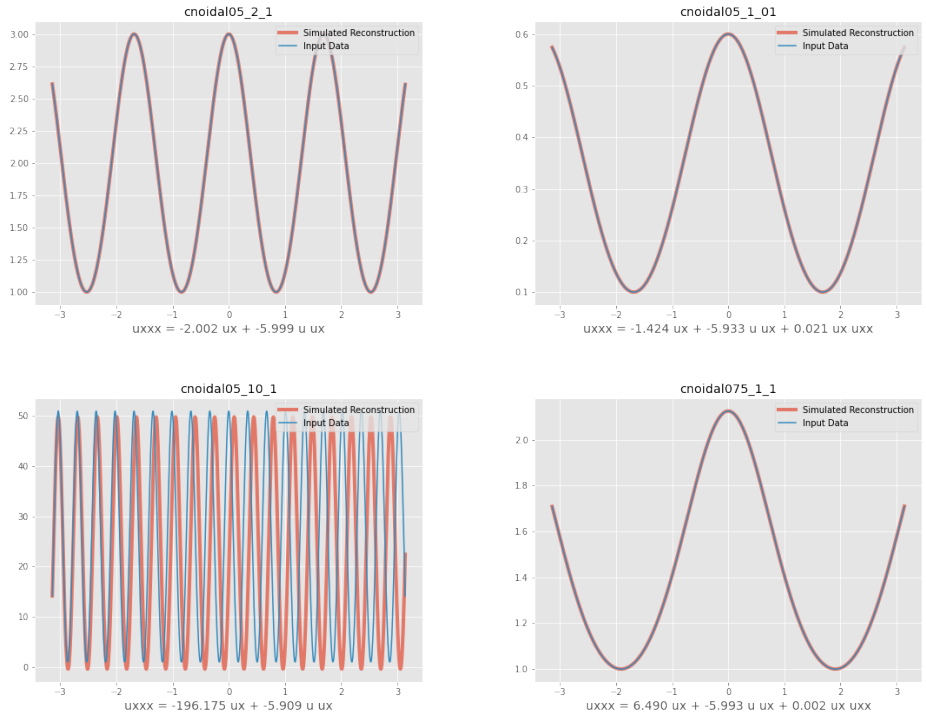

FIGURE 5. Cnoidal and predicted solutions to the KdV equation and the equations that they were drawn from. The x-axis shows the value of the traveling variable $x - ct$ at time zero, and the y-axis is the value of the solution at these locations. Each figure title shows the elliptic modulus k as the first number, followed by the width parameter γ and the minimal level of the solution b. For the waves on the left, SINDy identifies the stationary KdV equation, whereas on the right, it learns an extra term. In all cases the data is nearly fit.

This, in addition to the inconsistency of SINDY in identifying KdV from our KdV data, reflects that we may be unable to find the KdV equations from this stationary data alone, perhaps due to the fact that we are only giving SINDy one type of trajectory in phase space, a periodic or homoclinic orbit. Instead, we need to try to find time dependent solutions of the full water wave problem to analyze.

3.3. **Secondary results: time-dependent traveling waves.** We now apply SINDy to traveling waves in shallow water. Given traveling wave data $v_c(w)$ computed from the Babenko equation with wave speed c, we attempt to recover the time dependent dynamics by letting $w = x - ct$. To convert the data in time independent coordinates to time dependent data, we set $u(x,t) = v_c(x - ct)$ so that by the chain rule, $u_t = -cv_c'(x - ct)$ and

$u_x = v_c'(x - ct)$. Higher order derivatives in x can be obtained similarly. This yields data at one moment in time. Then, for each $t = t_0$, this gives a formula for $u(x, t_0)$ in terms of $v_c(w)$ for all x where $w_i = x_i - ct_0$. Setting

$$Y_c := \begin{pmatrix} u_t(x_1 - ct_0) \\ \vdots \\ u_t(x_m - ct_0)) \end{pmatrix} = \begin{pmatrix} -cv_c'(w_1) \\ \vdots \\ -cv_c'(w_m) \end{pmatrix},$$

$$X_c = \begin{pmatrix} u(x_1 - ct_0) & \cdots & u_{xxx}(x_1 - ct_0) \\ \vdots & & \vdots \\ u(x_m - ct_0) & \cdots & u_{xxx}(x_m - ct_0)) \end{pmatrix} = \begin{pmatrix} v_c(w_1) & \cdots & v_c'''(w_1) \\ \vdots & & \vdots \\ v_c(w_m) & \cdots & v_c'''(w_m) \end{pmatrix},$$

we apply the SINDy procedure to find a sparse solution to $Y_c = \Theta(X_c)\Xi$. Of course, there is one obvious flaw to this approach—since our data comes from a traveling wave, it can be perfectly modeled using one term on the right hand side, resulting in the advection equation. Indeed, because we assume that u is a traveling wave solution, this is the behavior we observe. Applying this procedure to traveling wave data at wave speeds 0.46 and 0.465, the equations we learn are $u_t = -0.46u_x$ and $u_t = -0.465u_x$, respectively. This is a good example of the fundamental unidentifiability issues that can be present in differential equation learning.

In order to get around this problem, we concatenate data from waves at multiple speeds together in the regression by setting $Y = (Y_{c_1}, Y_{c_2}, ..., Y_{c_k})$ and $X = (X_{c_1}, ..., X_{c_k})$. We regress using data from multiple traveling waves at once with the hope that our model will learn the map from wave shape to wave speed and use this to find a nonlinear PDE for which all of these solutions are traveling wave solutions at their respective wave speeds. If we set the sparsity parameter high enough such that only one term is remaining, we recover an equation of the form $u_t = -c^* u_x$ where c^* is some average of the different wave speeds. We apply this to data with (rounding to four digits) $c_1 = 0.1207$, $c_2 = 0.1500$, $c_3 = 0.1398$, $c_4 = 0.1278$, and obtain the equation $u_t = -0.1434u_x$. This model explains 99.8% of the variance u_t. Relaxing the sparsity requirement to obtain two terms on the right hand side, we can increase the

explained variance to 99.9%, and the differential equation found is (rounding to three digits),

$$u_t = -0.147u_x - 0.002uu_x,$$

which corresponds to the inviscid Burger's equation, a nonlinear wave equation where "taller" waves move faster. This matches our intuition about these traveling wave solutions: faster waves tend to be "pointier" and almost all of the variation in u_t is in the locations where the wave is pointy. Interestingly, this seems to suggest that it is difficult to capture dispersive behavior (u_{xxx} terms) from traveling wave solutions and that the next highest order term in the expansion around multiple traveling waves is the nonlinear uu_x term. This may potentially be related to the limited range of wave speeds for which we have waves to plug into SINDy: if we only have waves with a small range of potential speeds, then an advection equation with a speed equal to the average of all of the wave speeds models the data quite well. The problem of limited variance in the speed of the waves is inherent to getting traveling wave solutions in the fixed KdV parameter regime: to find waves in this regime, the waves must be calculated in very shallow water, and as the depth of the water decreases, the range of allowable speeds for periodic traveling surface water waves shrinks correspondingly.

4. Conclusion

In this work, we were able to use SINDy to reconstruct the Lorentz system as in [3]. We then investigated the feasibility of SINDy to be used as a model reduction technique, specifically for the full water wave problem. In particular, we aimed to see if SINDy would output the KdV equation as the governing equations given data simulated from the full water wave problem in the regime where the KdV equation well approximates the Euler water wave equations, i.e., for long waves in shallow water. If so, this would indicate that SINDy could serve as a viable model reduction technique; however, we found that SINDy could not recover the KdV equation effectively, even after performing cross validation. We then found that SINDy *could* recover the KdV equation given data simulated directly from the KdV equation itself. Lastly, we found that given cnoidal solutions to the full water

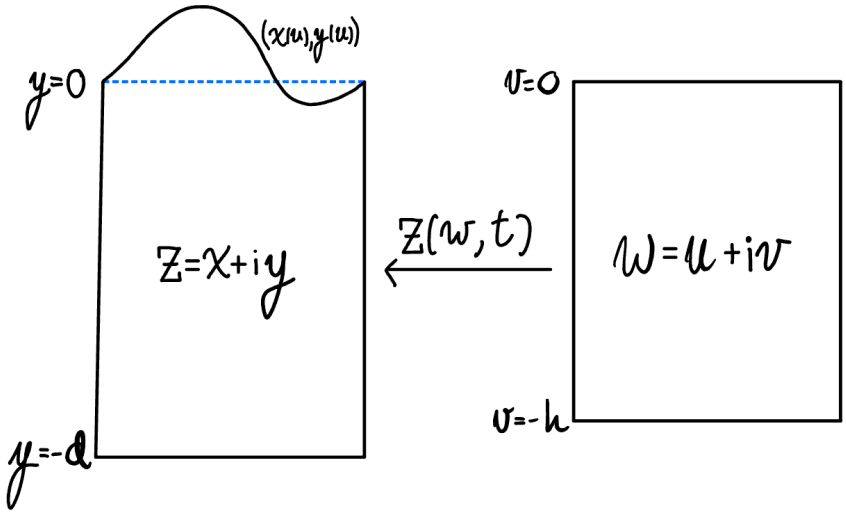

FIGURE 6. A diagram of the conformal mapping approach. On the left is the fluid domain which is mapped from a rectangular domain in the complex plane, shown on the right. This mapping is what is solved for in this approach.

wave problem in the KdV regime that SINDy could recover the KdV equation after cross-validation, but not consistently. As such, while SINDy has the potential to be used as a model reduction technique, in its current state, it cannot be reliably used to do such work.

APPENDIX A. DATA GENERATION

To apply SINDy to numerically simulated water wave data, we must first generate water wave data. To do this, we turn to the conformal mapping reformulation of the water wave problem as used in [9]. This reformulation allows us to replace four complicated differential equations with two much simpler ones at the cost of adding a few integral operators. It relies on the fact that solutions of Laplace's equation are mapped to solutions of Laplace's equation under conformal mappings of the complex plane, mapping a rectangle of points $w = u + iv$ in the complex plane onto the fluid domain $X + iY = Z$ such that $(X(u + i0), Y(u + i0))$ traces out the surface of the wave as u varies across its domain. It is then the mapping Z, in addition to the potential ψ along our surface, that we are solving for. One such mapping is shown in figure 6. The equations defining this reformulation are given by

$$Y_t X_u - Y_u X_t + \hat{R}\psi_u = 0,$$
$$X_t\psi_u - X_u\psi_t + \hat{T}(Y_t\psi_u - Y_u\psi_t) = g\left(X_u Y + \tilde{k}\left(\frac{Y^2}{2}\right)\right).$$

Here, $\hat{R}$ and $\hat{T}$ are generalizations of the Hilbert transform and its inverse. These are applied in Fourier space by scaling the kth Fourier mode as

$$(\hat{R}f)_k = i\tanh(hk)f_k, \quad (\hat{T}f)_k = -i\coth(hk)f_k.$$

Furthermore, the operator $\tilde{k}$ is defined as the composition of $\hat{T}$ with a spatial derivative

$$\tilde{k}f = \hat{T}\partial_u f.$$

We search for traveling wave solutions, as these are the type of solutions that first led to the derivation of KdV. To do this, we look for stationary solutions in a frame traveling with speed c to the left. This reduces our problem to solving just one equation, the so called Babenko equation:

$$\hat{S}y = c^2\tilde{k}y - g((1 - \hat{T}y_u)y + \hat{T}(yy_u)) = 0.$$

To access waves in the KdV regime, we need to generate waves which are both quite small but are also in very shallow water. We know the form of asymptotically small periodic traveling surface water waves exactly: they have the profile of a cosine wave. This only holds when the waves are asymptotically smaller than any of the other parameters in our equations, but it provides a good place to start. Once we have one traveling solution, we can generate more solutions of either different amplitudes or depths by perturbing either the speed at which our frame is traveling or the depth of the water which are baked into the Babenko equation. Then, we can use the previous solutions as initial conditions in a root-finding method for the perturbed equation. Specifically, we follow [7] by using a Newton-Conjugate Residual method and applying all operators spectrally.

This method gives traveling waves in whatever fixed depth and amplitude we want; however, generating more complicated solutions, such as a two-soliton solution or the evolution of generic initial data, is much more complicated with this model. There are a few reasons

for this. The first is that the application of the operators in Fourier space requires periodicity of the data: if the initial data is not periodic, this will very quickly lead to instabilities. The second issue is that while we can generate waves for whatever fixed depth we want in the conformal variables $w = u + iv$, this depth is not the same as the physical depth of the water we are trying to simulate. Specifically, if w belongs to a box shaped-domain of height h in the conformal domain and is used to generate a wave with surface height given by $y(u)$, Dyachenko and Hur proved that the physical depth d of that wave will be given by

$$d = h - \langle y \rangle,$$

where $\langle f \rangle$ is the average of a function f [6]. As a wave evolves in time, its average level could easily change, leading to the physical depth of the water changing. Furthermore, it is known that when given a single traveling wave as time-dependent data, SINDy will not correctly identify the PDE it came from, unless that PDE happens to be the advection equation. This is because the advection equation is the sparsest PDE that generates traveling waves, and with appropriate parameters, it can generate any wave traveling at a single speed. Brunton and Kutz noted this in their original paper [3]; however they were still able to identify KdV. They could do this by utilizing the fact that for KdV, solitary and periodic waves travel at a speed related to their amplitude and width. Thus, they could take a solution to KdV with two solitary waves traveling at different speeds and KdV would identify that.

One could also set up a similar initial condition for the wave surface and evolve it, however there are a few issues with this. The periodicity of traveling waves for KdV is unfortunately also dependent on the speed of these waves, so creating a truly periodic initial condition with two waves traveling at different speeds is challenging. Furthermore, in solving for traveling waves, we were able to remove the dependence on the potential ϕ; however, in time evolution, we need to take this potential into account. For our initial data, we can use the formula that ϕ obeys in the traveling waves

$$c\hat{R}\phi_u = y_u.$$

Since our two waves are traveling at different speeds, this unfortunately becomes inconsistent. Thus, evolving our data forwards in time becomes impractical using this approach.

However, we can create time dependent solutions from our stationary solutions in the moving frame: if the wave profile moving at speed c is given by $y(u)$, we simply define $y(u,t) = y(u-ct)$. We can do this for multiple speeds in order to generate a set of waves traveling at different speeds use all of this data in parallel in the SINDy algorithm as our simulation of time dependent water wave data in the KdV regime.

Appendix B. Derivation of KdV

The following derivation of the KdV equation from the Euler water wave problem is adapted from [1, Section 4.1] and Bernard Deconinck's unpublished lecture notes on nonlinear waves [5].

The one-dimensional Euler equations are the well-accepted model of water waves. Note that the formulation of these equations assume that the dynamics inside the water is irrotational and inviscid. Further, the model ignores the effects of surface tension and assumes that the water has constant density. The equation for the surface of the water is given by $z = \zeta(x,t)$, assuming that the bottom of the water is at $z = -h$, and the velocity (u,w) is given by $(u,w) = \nabla\phi = (\phi_x, \phi_z)$. With all of this, the Euler eqution are given by the following free-boundary problem:

$$\begin{cases} \phi_{xx} + \phi_{zz} = 0, & \text{for } -h < z < \zeta(x,t), \\ \phi_z = 0, & \text{at } z = -h, \\ \zeta_t + \phi_x \zeta_x = \phi_z, & \text{at } z = \zeta(x,t), \\ \phi_t + g\zeta + \dfrac{1}{2}(\phi_x^2 + \phi_z^2) = 0, & \text{at } z = \zeta(x,t), \end{cases}$$

with $\zeta, \phi \to 0$ as $|x| \to \infty$.

Linearizing the above equations around the equilibrium solution $\zeta(x,t) = 0$ and $\phi(x,z,t) = 0$, we find the linear dispersion relationship to be

$$\omega^2 = gk \tanh(kh).$$

We now consider the case of long waves in shallow water. If $|k|$ is small so that $|k|h \ll 1$, then we find $\omega^2 = ghk^2$ or $\omega_\pm = \pm\sqrt{gh}|k|$. In this case the waves are long compared to the depth of the water.

We introduce two small parameters, ϵ_1 and ϵ_2, that represent long waves with small amplitude in shallow water as follows:

$$\begin{cases} \epsilon_1 = (kh)^2 \ll 1 \\ \epsilon_2 = |\zeta|_{\max}/h \ll 1. \end{cases}$$

By Kruskal's principle of maximal balance, we let $\epsilon = \epsilon_1 = \mathcal{O}(\epsilon_2)$, and continue working with one small parameter ϵ only.

We define new dimensionless variables as follows: $z^* = z/h$, $x^* = \sqrt{\epsilon}\frac{x}{h}$, $t^* = \sqrt{\frac{\epsilon g}{h}}t$, $\zeta = \epsilon h\zeta^*$, and $\phi = h\sqrt{\epsilon gh}\phi^*$. We can then rewrite the Euler equations in terms of the nondimensionalized variables as follows,

$$\begin{cases} \phi_{zz} + \epsilon\phi_{xx} = 0, & \text{for } -1 < z < \epsilon\zeta, \\ \phi_z = 0, & \text{at} \quad z = -1, \\ \epsilon\zeta_t + \epsilon^2\phi_x\zeta_x = \phi_z, & \text{at} \quad z = \epsilon\zeta, \\ \phi_t + \zeta + \dfrac{1}{2}\phi_z^2 + \dfrac{\epsilon}{2}\phi_x^2 = 0, & \text{at} \quad z = \epsilon\zeta, \end{cases}$$

where we note that we omit the stars from the nondimensionalized variables for the sake of simplicity.

We must now determine the dependence on z. We begin by expanding $\phi(x, z, t)$ as a series in $z + 1$:

$$\phi(x, z, t) = \sum_{n=0}^{\infty}(z+1)^n\phi_n(x, t).$$

Substitution in $\phi_{zz} + \epsilon\phi_{xx} = 0$ gives:

$$\phi_{n+2} = \frac{-\epsilon\phi_{nxx}}{(n+1)(n+2)}, \quad n = 0, 1, ...$$

We know that $\phi_z = 0$ at $z = -1$, thus $\phi_1(x,t) = 0$. Using the above recursion relation, we find that $0 = \phi_1(x,t) = \phi_3(x,t) = \phi_5(x,t) = ...$, and hence

$$\phi(x,z,t) = \phi_0(x,t) - \epsilon\frac{(z+1)^2}{2!}\phi_{0xx} + \epsilon^2\frac{(z+1)^4}{4!}\phi_{0xxxx} - ...$$

Two equations remain to be solved, both at the surface $z = \epsilon\zeta(x,t)$:

$$\begin{cases} \phi_z = \epsilon\zeta_t + \epsilon^2\phi_x\zeta_x, & (3) \\ \phi_t + \zeta + \dfrac{1}{2}\phi_z^2 + \dfrac{\epsilon}{2}\phi_x^2 = 0. & (4) \end{cases}$$

Using our above results, we find that $\zeta_t + \partial_x\phi_{0x} = \mathcal{O}(\epsilon)$ from the first equation and $\phi_{0t} + \zeta = \theta(\epsilon) \Rightarrow (\phi_{0x})_t + \partial_x\zeta = \mathcal{O}(\epsilon)$ from the second equation.

We assume the following perturbation expansions

$$\phi_{0x} = u_0 + \epsilon u_1 + \epsilon^2 u_2 + ...,$$

$$\zeta = \zeta_0 + \epsilon\zeta_1 + \epsilon^2\zeta_2 + ...$$

Next, we introduce the slow time scales τ_1,τ_2, etc.:

$$\tau_0 = t, \tau_1 = \epsilon t, \tau_2 = \epsilon^2 t, ...,$$

which implies that

$$\frac{\partial}{\partial t} = \frac{\partial}{\partial \tau_0} + \epsilon\frac{\partial}{\partial \tau_1} + \epsilon^2\frac{\partial}{\partial \tau_2} + ...$$

We can then apply the perturbation expansions and slow time scale to equations (3) and (4). After some simplification, at first order we get that

$$\begin{cases} \dfrac{\partial^2\zeta_0}{\partial\tau_0^2} - \dfrac{\partial^2\zeta_0}{\partial x^2} = 0, \\ \dfrac{\partial^2 u_0}{\partial\tau_0^2} - \dfrac{\partial^2 u_0}{\partial x^2} = 0. \end{cases}$$

Both ζ_0 and u_0 satisfy wave equations with (dimensionless) velocity 1 as functions of x and τ_0. Thus

$$\begin{cases} \zeta_0 = f(x-\tau_0, \tau_1, \tau_2, ...) + g(x+\tau_0, \tau_1, \tau_2, ...), \\ \\ u_0 = f(x-\tau_0, \tau_1, \tau_2, ...) + g(x+\tau_0, \tau_1, \tau_2, ...). \end{cases}$$

Thus, on the fastest time scale τ_0, all initial data splits in two directions, with one part (f) propagating to the right, and the other part (g) propagating to the left.

Let us proceed to higher order. First we determine $\phi_z(x, \epsilon\zeta(x,t), t)$ to second order in ϵ, which results in

$$\phi_z(x, \epsilon\zeta(x,t), t) = -\epsilon u_{0x} + \epsilon^2 \left(-\zeta_0 u_{0x} - u_{1x} + \frac{1}{6} u_{0xxx} \right) + \mathcal{O}(\epsilon^3).$$

Next, we determine $\phi_x(x, \epsilon\zeta(x,t), t)$ to first order in ϵ to get that

$$\phi_x(x, \epsilon\zeta(x,t), t) = u_0 + \epsilon \left(u_1 - \frac{1}{2} u_{0xx} \right) + \mathcal{O}(\epsilon^2),$$

and $\phi_{xt}(x, \epsilon\zeta(x,t), t)$, also to first order in ϵ to get

$$\phi_{xt}(x, \epsilon\zeta(x,t), t) = u_{0_{\tau_0}} + \epsilon(u_{0_{\tau_1}} + u_{1_{\tau_0}} - u_{0xx_{\tau_0}}/2) + \mathcal{O}(\epsilon^2).$$

Last, we get the following expression for ζ_t to first order in ϵ

$$\zeta_t = \zeta_{0_{\tau_0}} + \epsilon(\zeta_{1_{\tau_0}} + \zeta_{0_{\tau_1}}) + \mathcal{O}(\epsilon^2).$$

Using all of these expressions, we find that the kinematic and Bernoulli equations to second order in ϵ become

$$\begin{cases} \zeta_{1_{\tau_0}} + u_{1_x} = -\left(\zeta_{0_{\tau_1}} + u_0 \zeta_{0_x} + \zeta_0 u_{0_x} - \frac{1}{6} u_{0xxx} \right), \\ \\ u_{1_{\tau_0}} + \zeta_{1_x} = -\left(u_{0_{\tau_1}} - \frac{1}{2} u_{xx_{\tau_0}} + u_0 u_{0x} \right). \end{cases}$$

Let us change variables on these two equations, to express the in characteristic variables. We introduce variables that "diagonalize" the system of equations on the left-hand side:

$$l = x + \tau_0, \quad r = x - \tau_0.$$

Here, l is the characteristic variable for a left-moving wave and r is the characteristic variable for a right-moving wave. Then

$$\partial_x = \partial_r + \partial_l, \quad \partial_{\tau_0} = \partial_l - \partial_r,$$

and $\zeta_0 = f(r) + g(l)$, $u_0 = f(r) - g(l)$. The transformed equations are

$$\begin{cases} \zeta_{1l} - \zeta_{1r} + u_{1r} + u_{1l} = -\left(f_{\tau_1} + g_{\tau_1} + (f-g)(f_r + g_l) + (f+g)(f_r - g_l) - \frac{1}{6}(f_{rrr} - g_{lll}) \right), \\ u_{1l} - u_{1r} + \zeta_{1r} + \zeta_{1l} = -\left(f_{\tau_1} - g_{\tau_1} + \frac{1}{2}(f_{rrr} + g_{lll}) + (f-g)(f_r - g_l) \right). \end{cases}$$

Adding and subtracting these two equations and integrating the left-hand sides results in

$$\begin{cases} 2(\zeta_1 + u_1) = -\left(2f_{\tau_1} + 3ff_r + \frac{1}{3}f_{rrr} \right) l + f_r \int g \mathrm{d}l + \frac{1}{2}g^2 + fg - \frac{2}{3}g_{ll} + C_1, \\ 2(u_1 - \zeta_1) = -\left(2g_{\tau_1} - 3gg_l - \frac{1}{3}g_{lll} \right) r - g_l \int f \mathrm{d}r - \frac{1}{2}f^2 - gf + \frac{2}{3}f_{rr} + C_2, \end{cases}$$

where C_1 (C_2) may depend on all variables but l (r). All terms on the right-hand sides are bounded, except for the first terms which are secular and the second terms (unless the integrand has zero average). The second terms of both equations may be eliminated by imposing that we consider profiles for which the right- and left-translating frames of reference do not interact. Thus, the equations we are about to derive are valid in frames moving along with the shallow-water velocity. This leaves only the growth of the first secular terms. This growth is unphysical, as the quantities on the left-hand side are clearly bounded. Thus, we impose that the secular terms vanish. This gives

$$2\frac{\partial f}{\partial \tau_1} + 3f\frac{\partial f}{\partial r} + \frac{1}{3}\frac{\partial^3 f}{\partial r^3} = 0,$$

for the right-going wave, and

$$2\frac{\partial g}{\partial \tau_1} - 3g\frac{\partial g}{\partial l} - \frac{1}{3}\frac{\partial^3 g}{\partial l^3} = 0,$$

for the left-going wave. We have obtained two PDEs satisfied by f and g. Both are KdV equations. Thus, in their respective frames of reference, both parts of the wave profile are

described by a KdV equation which describes how the right- or left-going part of the profile evolves in its moving frame of reference.

Appendix C. Scaling transformation of KdV

Let $u(x,t)$ be a solution to $u_t = uu_x + u_{xxx}$. We want to show that the addition of arbitrary coefficients is irrelevant so long as suitable scalings of both the independent and dependent variables are chosen.

Let the arbitrary coefficients be denoted by α and β. The KdV equation then becomes

$$u_t = \alpha u u_x + \beta u_{xxx}. \tag{5}$$

We note that we do not place a coefficient in front of the u_t term, because this coefficient is always able to be divided out.

We now apply the following linear scalings to all of the variables in the KdV equation:

$$X = Ax,$$

$$T = Bt,$$

$$U = Cu.$$

Applying the chain rule to our scalings yields:

$$\frac{\partial}{\partial x} = \frac{\partial X}{\partial x}\frac{\partial}{\partial X},$$
$$\frac{\partial}{\partial t} = \frac{\partial T}{\partial t}\frac{\partial}{\partial T}.$$

Applying our scalings to equation (5) yields

$$\frac{\partial(Cu(Ax,Bt))}{\partial t} = \alpha Cu(Ax,Bt)\frac{\partial(Cu(Ax,Bt))}{\partial x} + \beta\frac{\partial^3(Cu(Ax,Bt))}{\partial x^3}.$$

Applying our chain rule to this yields:

$$CB\frac{\partial u}{\partial T} = \alpha C^2 Au\frac{\partial U}{\partial X} + \beta CA^3\frac{\partial^3 u}{\partial x^3}.$$

Dividing through by CB yeilds:

$$\frac{\partial u}{\partial T} = \frac{\alpha CA}{B} u \frac{\partial u}{\partial X} + \frac{\beta A^3}{B} \frac{\partial^3 u}{\partial X^3}. \tag{6}$$

We have complete control over our scaling parameters. Thus, we can choose A, B, and C such that the coefficients $\frac{\alpha CA}{B}$ and $\frac{\beta A^3}{B}$ both equal 1. We note that this is possible due to the fact that $\frac{A}{B}$ and $\frac{A^3}{B}$ have the same sign. Thus, the third parameter, C, gives us the ability to ensure that both coefficients can be 1 regardless of the values of α and β. We note however, that we have two equations and three unknowns. As such, we let A be some nonzero value, λ. Setting this up yields:

$$\frac{\alpha CA}{B} = 1 \qquad \frac{\beta A^3}{B} = 1.$$

Applying $A = \lambda$:

$$\frac{\alpha C\lambda}{B} = 1 \qquad \frac{\beta \lambda^3}{B} = 1 \Rightarrow B = \beta\lambda^3.$$

Plugging in our expression for B:

$$\Rightarrow \frac{\alpha C\lambda}{\beta\lambda^3} = 1 \Rightarrow C = \frac{\beta\lambda^2}{\alpha}.$$

Thus, with $A = \lambda$, $B = \beta\lambda^3$, and $C = \frac{\beta\lambda^2}{\alpha}$, and a clever choice of the nonzero value, λ, we can choose the coefficients of equation (6) to be 1 regardless of the values of α and β. Therefore, equation (6) becomes

$$u_T(X,T) = u(X,T)u_X(X,T) + u_{XXX}(X,T). \tag{7}$$

Because $u(x,t)$ is a solution to $u_t(x,t) = u(x,t)u_x(x,t) + u_{xxx}(x,t)$, then $u(X,T)$ is a solution to equation (7). As such, we have shown that by choosing a suitable scaling, we can use any coefficients in the KdV equation that we please.

Appendix D. Code

All code utilized in this paper can be found at [2].

References

[1] M. J. Ablowitz and H. Segur. *Solitons and the Inverse Scattering Transform*. Society for Industrial and Applied Mathematics, 1981.

[2] C. Ballew, E. Byrnes, A. Hsu, S. Ichinaga, and K. Lilly, 2023. `https://github.com/sichinaga/amath575_project`.

[3] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, March 2016.

[4] B. M. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, 2020.

[5] B. Deconinck. Nonlinear Waves. 2022.

[6] S. A. Dyachenko and V. M. Hur. Stokes waves with constant vorticity: I. Numerical computation. *Studies in Applied Mathematics*, 142(2):162–189, 2019.

[7] S. A. Dyachenko, P. M. Lushnikov, and A. O. Korotkevich. Branch cuts of Stokes wave on deep water. Part I: numerical solution and Padé approximation. *Studies in Applied Mathematics*, 137(4):419–472, 2016.

[8] A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton. PySINDy: A comprehensive Python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, 2022.

[9] V. E. Zakharov, E. A. Kuznetsov, and A. I. Dyachenko. Dynamics of free surface of an ideal fluid without gravity and surface tension. *Fizika plasmy*, 22:916–928, 1996.