Learning PDEs from Data: Application to Surface Engineering

Erick Gomez Milan Jankovic Mohamed Moujahed Hedi Zeghidi

Abstract

The lack of register in images from a laboratory experiment undermines the search for a PDE that can explain the underlying dynamics. We propose two alignment methods to solve the absence of register, and a sparse regression method to estimate the PDE. The first alignment method is the R2 score alignment, based on successive linear regressions and similarity. The second method is the Genetic Alignment, and it uses entropy instead of similarity. After aligning the images, we compare the results from *Sindy*, a known method in Physics Informed Machine Learning, and the sparse regression method we propose, called Iterative Huber Ridge. The advantage of the sparse regression method we propose is the capacity to deal with outliers or misalignment. Our method finds derivative terms in the PDE, unlike *Sindy*, and has lower Relative Mean Absolute errors.

1. Introduction

The purpose of this document is to find a PDE that represents the dynamics in an experiment on Nickel after laser exposure, and the resulting self-organization of particles (Nakhoul et al., 2021). Specifically, we study self-organization in the context of the nanostructuring process induced by ultrafast laser irradiation on nickel surfaces. In our case, we continue the study of the spontaneous formation of complex nanostructures on the nickel surface as a result of the interaction with the femtosecond laser.

The main obstacle to find a PDE that describes the dynamics of the Nickel under the influence of laser is the absence of register in the images. This means we ignore which pixel at an image corresponds to the following image. Consequently, it is not possible to calculate the derivative of the pixel intensity with respect to time.

One of the subbranches of Physics Informed Machine Learning focuses on the discovery of PDEs from synthetic or experimental data. In the experiment we study, it is possible to calculate the derivatives of pixel intensity with respect to the x and y axis, in several orders, and also to create new features by forming monomials with the derivatives. From this large library of features, sparse regression methods provide a concise PDE with low errors.

Before putting into practice sparse regression, we must align the images. We propose two methods. Further details are in Section 4. Briefly, the first method calculates a linear regression using as predictors a small set of derivatives. We crop a patch from an image and for every other possible patch in the next image we calculate the R2 score. We select the patch that shows the highest R2 score. The default of this method, and of methods based on similarity, is the risk of losing advection, we call this risk the Alignment Trap.

As an illustration of the Alignment Trap, imagine a wave that only moves, does not suffer deformation. Imagine we pick a patch A at a point in time, and we want to find a patch B at the next point in time. We want patches that align. If we look for patch most similar to patch A, we will come out with a patch B identical to patch A. Therefore, in the quest for similarity we lose dynamics.

The second method of alignment avoids the alignment trap. Instead of similarity it uses entropy. Self-organization implies a reduction in entropy and later a stabilization of entropy. In a sequence of misaligned patches, we can calculate the entropy for each patch and it reveals a noisy entropy line or path. We smooth this entropy to approximate the unobserved entropy path. A sequence of patches that closely follow this approximated entropy is the sequence of aligned patches. We explore the space of all possible sequences using a genetic algorithm to find a sequence of patches whose entropy resembles the smoothed entropy path.

After alignment of the images, the main sparse regression method proposed by Rudy et al. (*trainSTRidge*) shows higher variability in the number of coefficients of the PDE and higher errors than the Iterative Huber Ridge.

The document follows this order. After the literature review on sparse regression, we show the perils of misalignment applying sparse regression of synthetic data. Section 3 describes the data. In Section 4, we explain the two alignment methods and the Iterative Huber Ridge method for sparse regression. Section 5 describes the setup. Section 6 and 7 show the results and discuss future research paths. Finally, the conclusions are in Section 8.

2. Literature Review

Physics-informed machine learning (PIML) has emerged as a powerful approach for discovering partial differential equations (PDEs) using sparse regression methods, particularly the Sparse Identification of Nonlinear Dynamics (SINDy) framework (Rudy et al., 2017). The SINDy framework has been widely utilized for the data-driven discovery of PDEs, although it is dependent on the quality and quantity of measurement data due to the requirement of numerical differentiation for computing derivatives (Chen et al., 2021). The original SINDy framework was designed to recover systems of ordinary differential equations (ODEs) and shows good performance recovering PDEs as well (Bertsimas & Gurnee, 2023).

The integration of SINDy with Poincaré Maps (tool used to analyze the behavior of a dynamical system by reducing the dimension of its phase space by one) was shown as a robust method to discover multiscale nonlinear dynamics, showing the versatility of the SINDy framework (Bramburger & Kutz, 2020). Additionally, a new method named Sparse Identification of Nonlinear Dynamics with Levenberg-Marquardt algorithm (SINDy approach is used to identify the nonlinear dynamical systems, and the Levenberg-Marquardt algorithm is employed to enhance the accuracy of the mathematical model) has been employed for modeling and predicting the transmission dynamics of COVID-19, highlighting the broad applicability of SINDy in studying nonlinear systems (Jiang et al., 2021).

Moreover, the SINDy framework has been extended to identify implicit dynamics and rational nonlinearities, leading to the development of SINDy-PI (a robust variant of the SINDy algorithm developed to identify implicit dynamics and rational nonlinearities in dynamical systems described in (Kaheman et al., 2020)). Additionally, the Weak SINDy (variant of SINDy algorithm that utilizes test functions to access derivative data through integration by parts) method has been extended for recovering PDEs from data, demonstrating the adaptability of the SINDy framework to various types of differential equations (Messenger & Bortz, 2021).

Among the regression techniques used in PDE discovery is the Huber regression (Huber, 1992). The huber loss is designed to be less sensitive to outliers than the mean squared error (MSE) loss. The Huber loss combines characteristics of both the mean absolute error (MAE) and the mean squared error. Long et al. implements a neural network to identify the derivative terms in a PDE. The loss function for the neural network is the Huber loss. Later, Chen et al. on the same line, improves the method to identify the partial derivatives. Another approach is to train a neural network as the surrogate for a solver and, also, use the huber loss (Pestourie et al., 2023).

3. Data

3.1. Experimental Data

The images come from an experiment described by Nakhoul et al.. They used a mono-crystalline Nickel (Ni) sample oriented in (100) direction, which was cut into $10 \times 10 \times 10$ mm cubes. The surfaces were polished in two ways: mechanical and electrochemical polishing. The polishing procedures were aimed at achieving high-quality surfaces with minimal roughness. Polishing is crucial for studying the effects of laser irradiation on nanoscale surface topography. The sample was then irradiated with a Titateneum:sapphire laser. It is a popular type of tunable laser that emits short pulses of intense light in the near-infrared part of the electromagnetic spectrum. Different pulse duration and output power of the laser had different effects.

Surface topography was visualized using scanning electron microscopy (SEM) and atomic force microscopy (AFM), and roughness calculation was performed using NanoScope Analysis software. The study observed the formation of nanopeaks, nanobumps, hexagonal self-organized nanohumps, and nanocavities on the surface.

The nanopeaks and nanobumps were characterized by small, isolated protrusions on the surface, while the hexagonal self-organized nanohumps and nanocavities formed larger, more complex structures. The hexagonal self-organized nanohumps were arranged in a hexagonal pattern, with a height of tens of nanometers and a diameter of around 30 nm. The nanocavities, on the other hand, were characterized by a depression in the surface, with a diameter of around 30 nm and a depth of tens of nanometers.

The laboratory experiment provides sequences of images of size 1920 by 2200 pixels. Each sequence has 50 images and associated Fluence and Delay parameters. We focus on the experiment with Fluence equal to 0.20 and Delay equal to 2 (see Fig. 1). Furthermore, we align patches of size 300 by 300.

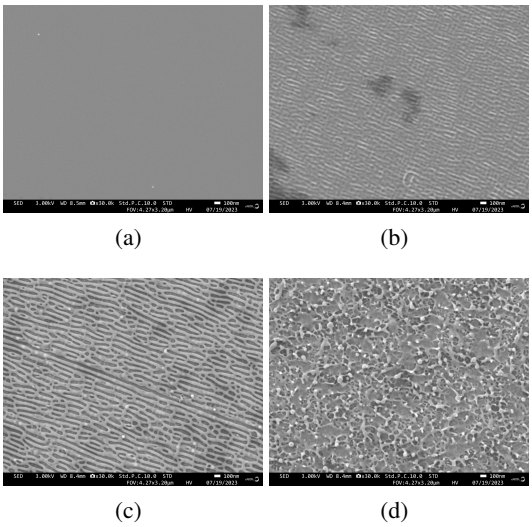
3.2. Synthetic Data

We generate synthetic data from a PDE and a solver implemented in Python for the equation:

$$\frac{\partial w}{\partial t} = rw - w^2 - s\cos\nabla w$$

Where

- w is the spatio-temporal pattern, representing the deviation from a homogeneous state.
- r is a parameter controlling the linear growth rate.
- s is a parameter controlling the strength of the nonlinear term



(a) (b) (c) (d)

Figure 1. Laboratory experiment using a mono-crystalline Nickel (Ni) sample oriented in (100) direction, which was cut into 10 × 10 × 10 mm cubes. The surface was exposed to laser with Fluence equal to 0.20 and Delay equal to 2 at timestep (a) 1 (b) 10 (c) 20 (d) 40

- ∇^2 is the Laplacian operator, representing spatial diffusion.
- rw captures linear growth.
- $-w^2$ represents self-repression.
- $\nabla^2 w$ captures spatial diffusion.
- $s \cos \nabla w$ is a non-linear interaction term for pattern formation.

The parameters r and s, in the equation above, take values $(0.5, 2), (0.1, 2.2), (0, 2), (0.5, 1)$. We name each set of generated synthetic data A, B, C, and D, respectively.

If we pick a pixel from centered patches belonging to the synthetic data B, and we plot the change of pixel density in the same i, j coordinates, we observe smooth change. However, if we take a random alignment of patches and plot the evolution of pixel intensity, we observe higher variability and for some time periods, different trend (see Fig 4).

In Fig 16, we create a random misalignment of patches for each synthetic data set A,B,C, and D. The dotted orange points are the entropy of centered patches, i.e. aligned patches. The blue line is the Lowess approximation from the entropy values of the misaligned patches. The rows in Fig 16 show different Lowess curves for the fraction of values the smoothing takes in local approximation. We observe that the lowess curve is a good approximation to the ground truth for low fraction values in the case of the B data set.

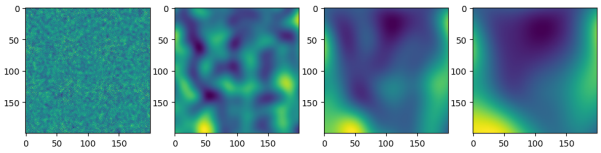


Figure 2. Synthetic data set B. Patches from a ground truth alignment.

In the Fig.3 and 2, we observed the patches at time 0,1,5,10 of the synthetic data with two different methods, the first one is the ground truth alignment we want and the second one is a random alignment.

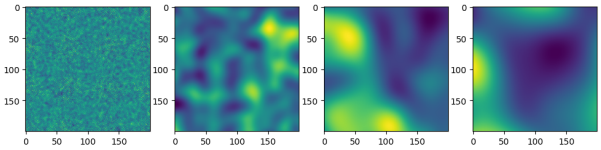


Figure 3. Synthetic data set B. Patches from a random alignment.

In the Fig.4, we pick one pixel and plot the evolution of the value of the pixel (in particular the derivative wrt to time). The orange line comes from well aligned patches and the line is smooth. The blue line comes from misaligned patches and it is not smooth. and it even has a different trend for some time periods.

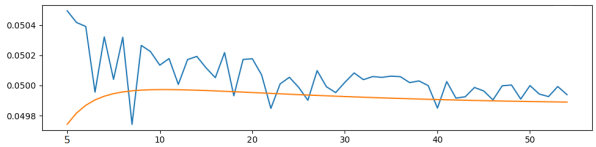


Figure 4. Pixel evolution with in orange the ground-truth alignment and in blue the random alignment

4. Methodology

4.1. R2 score alignment

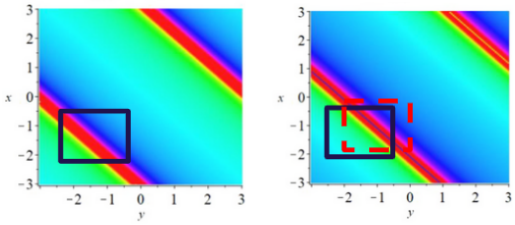
We pick a patch in an image at time point t. We create a set of every possible patch in an image at time point $t + 1$. Since the images show periodicity, we restrict the search to the upper left quadrant of the images. We create pairs of patches, where the first element is the patch in the first image, and the second element is a patch from the set of patches that come from the second image.

For each pair of patches, we calculate an approximation to dw/dt as the difference of pixel intensity. Using information on the second patch, we create a small library of derivatives w_x, w_y, w_{xx}, w_{yy}, the pixel value on the second patch w, w^2, w^3, and the intercept.

A measurement of linear regression fit that shows the proportion of explained variance is the R2 score, equal to $1 - \frac{\text{SSR}}{\text{SST}}$. The range of possible values is $(-\inf, 1]$. In our case, 1 means perfect alignment.

4.2. Alignment trap

The main issue with the R2 method is the risk to fall into the Alignment Trap. Given two images at different time points (such as in the Fig.5). If we pick a patch from one image and we look for the patch giving the best r2 score in the neighbour of the next image, we might lose the movement of the wave. The PDE we can find with this data is extremely sparse, it is just the constant zero, and the R2 score from the alignment of these two patches is equal to one, a perfect alignment.



(a) (b)

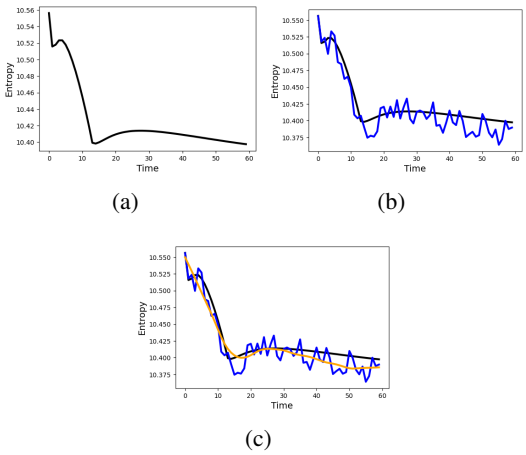
Figure 5. Image at timestep (a) 0 (b) 1

4.3. Genetic alignment

We introduce a novel alignment technique, Genetic Alignment. In demonstrating its efficacy, we utilize synthetic data showcased at the beginning of the report. By selecting properly aligned patches, we compute the entropy for each, resulting in the creation of a black curve (in the Fig.6(a)). Conversely, when calculating entropy for misaligned patches, we generate a blue curve (in the Fig.6(b)). Utilizing the data from the blue curve, we employ a Lowess (locally weighted Scatter Smoothing) approximation, essentially smoothing the curve (in the Fig.6(c)).

Remarkably, even with a different set of misaligned patches, the resultant blue curve changes, yet the Lowess approximation remains stable and closely mirrors the black curve, representing the true entropy path. This observation sparks an intriguing idea: searching for patch alignments that exhibit an entropy path akin to the Lowess approximation.

The number of possible sequences of patches of size 300 by 300 is approximately $4.8e56$, a quantity greater than the number of possible chess moves. Although, periodicity lowers the number, it is still a higher bound. The exploration of large space of solutions requires a different approach. We apply a Genetic Algorithm (Holland, 1992), whose principles remind evolution in biology: selection, crossover (recombination), and mutation to evolve a population of potential solutions to approach the solution to a problem over successive generations.



(a) (b)

(c)

Figure 6. Image at timestep (a) 0 (b) 1

The genetic algorithm starts with a population of 5000 candidate solutions or alignments, we create these candidates randomly. We evaluate how close the entropy path of each of these candidates follow the lowess approximation with the mean squared error (metric used to measure the average of the squares of errors or deviations between the estimated values and the actual values). Then we create the new generation by two methods:

1. pick parents to create offspring, the parents at 80% in best candidates (top 40%) and 20% the rest of the population. Regarding recombination, the offspring are created in two ways. First, the new child is the average of each alignment of the parents. Second, we create two children by doing a crossover such as in the Fig.7, where we combine 50% of the first alignment of parent one and 50% of the last alignment of parent two, and also the inverse to create another new alignment

2. mutate candidates (the candidates are selected at the beginning of the genetic algorithm with the all population but in the later generation we picked in the candidates with the smaller MSE) by randomly moving one alignment of one image by a certain number

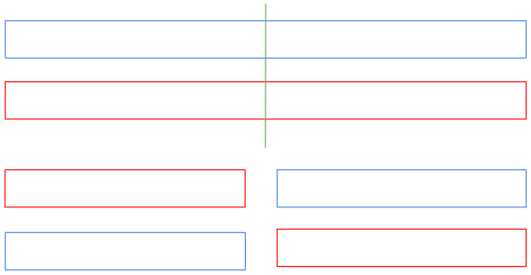
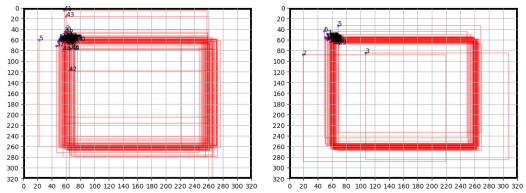


Figure 7. Creation of offspring by crossover of two parents

We use our genetic algorithm during 800 generations on the synthetic data B and D and we obtain on the last generation with the smallest MSE the alignment in the Fig.8. On the left and right, the 60 patches selected are mostly concentrated in the center of the image.



(a) (b)

Figure 8. (a) B (b) D

Now, we use it on the experimental data with a Fluence 0.20 and Delay 2 in the format as the synthetic data and we obtain the alignment below in the Fig.9 with the different patches also concentrated in the center of the images.

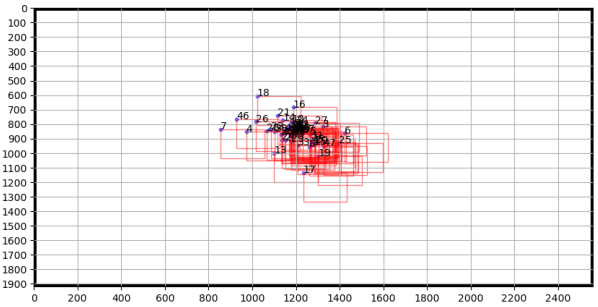


Figure 9. Best alignment selected by Genetic alignment after 800 generation for the experimental data with the Fluence 0.22 and Delay 2

4.4. Sparse regression

A detailed explanation of the main regression algorithm used in the Sindy framework is in (Rudy et al., 2017). Here we present a brief description. The algorithm is called *trainSTRidge* and it consists a two loops: an outside loop, and an inside loop, which is nested in the first. Given a library of features (derivatives and polynomials from the derivatives), *trainSTRidge* splits the sample in test and train sets. Using the test set, a linear regression with $L0$ penalty outputs its loss. The loop nested inside runs linear regressions with ridge penalty and leaves out the features whose coefficients are greater than the tolerance. This loop runs until convergence or to a maximum number of iterations.

Using the weights from the output of the nested loop, the *trainSTRidge* algorithm calculates the loss as the sum of the errors and the $L0$ penalty in the test data. If the loss is lower than the value obtained before running the nested loop, the new weights are accepted. Else, the tolerance is adjusted. In either case, the inside loop runs again. The algorithm stops when the maximum number of iterations for the outside loop is reached. In short, the outside loop looks for lower errors without overfitting, and the inside loop looks for sparsity.

The sparse regression method we propose is the Iterative Huber Ridge sparse regression. The goal is to consider the possibility of misalignment even after employing an alignment method. The Huber loss gives less influence to outliers by using a range $[-delta, delta]$ of errors where it squares them, and outside of the range it takes the absolute value. We employ a Huber loss that auto-tunes the parameter $delta$ to pick as outliers a given proportion of the sample. The total loss is the sum of the squared errors of the non-outliers, the absolute errors of the outliers, and the Ridge penalty to avoid large coefficients in the PDE. We apply this regression to a maximum number of iterations or convergence. At the end of each iteration we compare the value of each coefficient to a tolerance level. If the coefficient is smaller, we take out the corresponding feature from the matrix of features.

Below, the equation of the Huber loss divides the treatment of the errors according to its size in comparison to δ.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta \\ \delta(|y - f(x)| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$

5. Setup

Library of features. The library of features is composed by the partial derivatives with respect to the x and y, from the first derivative to the fourth derivative, and the cross derivatives. The regression captures the nonlinearities through monomial terms formed by the partial derivatives. The degree of the monomials is one, two, or three.

Grid search The *trainSTridge* algorithm has five hyperparameters: tolerance, L0 penalty, L2 penalty, maximum number of iterations in the main loop, and maximum number of iterations in the small loop. Regarding the maximum

number of iterations, we take 25 and 10, respectively. The L0 is the conditioning number of the feature covariance matrix. We perform a grid search on the tolerance and L2 penalty.

$$\text{L2} = [1.e-05, 3.e-05, 5.e-05, 7.e-05, 9.e-05]$$

$$\text{Tolerance} = [25, 125, 250, 1400, 4000]$$

The *Iterative Huber Ridge* algorithm has four hyperparameters: maximum number of iterations, tolerance, and outlier proportion. The number of iterations is 10. Although, the algorithm converges after 4 iterations in most of the cases. The tolerance is between 0.45 and 0.25. When the algorithm fails to pick any feature, we reduce the tolerance. The outlier proportion is set to 10%.

Sample size Each of the rolling window regressions take 5% of the data as sample. In the case of *trainSTRidge* regression, the 5% is further split into train and test sets, of size 80% and 20%, respectevely. The Huber regression takes a sample of 10%. We increased the sample because the algorithm is faster than *trainSTRidge*.

6. Results

6.1. Results on synthetic data

We apply the *trainSTRidge* to the synthetic data B with parameters $r = 0.1, s = 2.2$. We consider sequences of patches. The first one is the centered patches. Therefore, the patches are aligned. In the second sequence we pick a patch at random for each image. In consequence, the sequence is misaligned. In Table 1, we show the errors from the two sequences of patches. All the types of errors are larger when the data is misaligned.

	MSE	MAE	RMAE
Aligned B	1.90e-5	0.0035	8.7%
Missaligned B	0.016	0.1014	12.0%

Table 1. Errors from *trainSTRidge* regression on synthetic data B.

The PDEs obtained by trainSTRidge using the synthetic data set B differ according to their alignment. When the patches are aligned, the PDE is sparse, and manages to represent the nonlinearity as $Cosine$ functions of w_x and w_y. When the patches are misaligned, we lose sparsity.

1. Aligned:$w_t = 0.492 * w_{yy} - 0.321 * cos(w_x) + 0.321 * cos(w_y) + 0.00023 * w$
2. Misaligned: $w_t = 0.0045 * w_x w_{xxx} - 0.0010 * w_{xx} w_{yy} + 0.1048 * w_{xx} w_{xxy} + 4.0226 * w_{xxx}^2 + 9.2604 * w_{xxx} w_{xxy} - 0.2064 * w_{xxxx} w_{yy} - 0.0036 * w_y w_{xxy} - 0.0125 * w_{xy}^2 + 6.9929e-10 * w + -7.5481e-10 * w^2 + 1.595e-10 * w^3$

6.2. Results on experimental data

Before proceding with the regression, we apply the R2 score alignment method to the images from the experimental data with Fluence of 20 and Delay of 2. In Fig 10, we show the R2 score for every possible patch in image at time $t = 6$, where we use the centered patch at time $t = 5$ to calculate dw/dt. We choose the patch in image $t = 6$ with the highest R2 score to be the next patch in the sequence. From this last patch, we repeat the same procedure to find the next one.

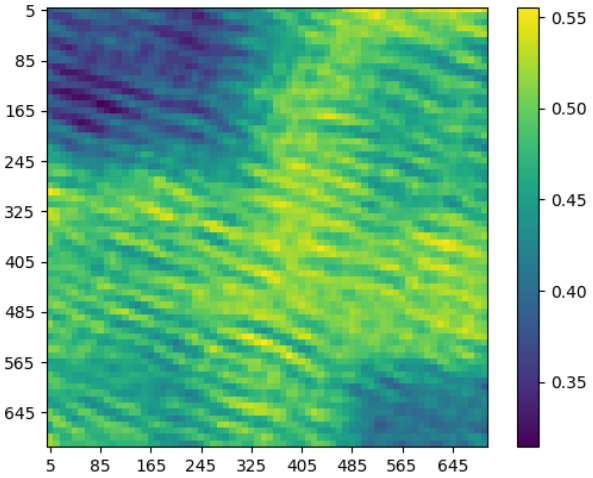


Figure 10. R2 score between a patch in t = 6, and each possible patch in t = 7

In Fig 11 we show the number of coefficients in each estimated PDE when we take a rolling window of size 4 in the R2 aligned patches from the experimental data. The regression algorithm is *trainSTRidge*. We observe high variability in the number of coefficients. Also, the Relative Mean Absolute Errors are for the most part above 10%.

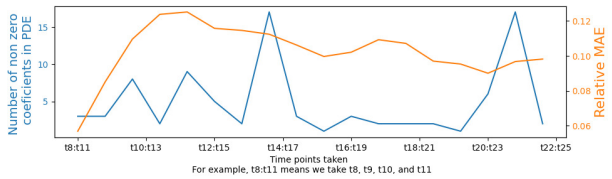


Figure 11. Left, number of terms, and RMAE of PDEs estimated using R2 alignment with trainSTRidge.

In Fig 12 we show the number of coefficients in each estimated PDE when we take a rolling window of size 4 in the R2 aligned patches from the experimental data. The regression algorithm is *Iterative Huber Ridge*. We observe lower variability in the number of coefficients, in comparison to Fig 11. The Relative Mean Absolute Errors that do not consider outliers are below 2%.

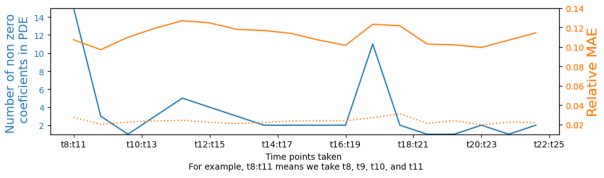


Figure 12. Left, number of terms, and RMAE of PDEs estimated using Iterative Huber Ridge. The dotted orange line does not consider outliers

We apply the two alignment methods (R2 and Genetic), and the two sparse regression methods (*trainSTRidge* and *Iterative Huber Ridge*, to the experimental data. The PDEs we find with *trainSTRidge* do not find terms with derivatives. While, the *Iterative Huber Ridge* method manages to pick terms with derivatives, as well as self-repression.

1. PDE of R2 alignment with trainSTRidge, $w_t = 1.20148003 * w - 4.62586101 * w^2 + 4.23625651 * w^3$
2. PDE of GA alignment with trainSTRidge, $w_t = -0.00165616 * w^2$
3. PDE of R2 alignment with IterativeHuberRidge, $w_t = 0.534 * w^2_{xxx} - 0.342 * w_y w_{yyy} + 0.437 * w^2_{yyy} + 4.064 * w^2_{yyyy} + 4.608 * w^2_{xxy} + 1.127 * w^2_{xxxy} + 5.047 * w^2_{xyy} + 1.397 * w^2_{xyyy} - 0.003 * w - 0.006 * w^2$
4. PDE of GA alignment with IterativeHuberRidge,$w_t = -1.2729 * w_{xxxx} w_{yy} - 1.0652 * w^2_{yy} - 1.912 * w^2_{xxyy}$

In Table 2, we show the errors from *trainSTRidge* and *Iterative Huber Ridge* in the experimental data aligned using either the R2 score or the Genetic Alignment method. The RMAE for both alignment and both regression methods is aroud 10.0%, however, the RMAE when we do not consider outliers in the case of the *Iterative Huber Ridge* method is below 4.0%.

	MSE	MAE	RMAE	MSE without outliers	MAE without outliers	RMAE without outliers
R2 aligned with STRidge	0.0052	0.0572	11.15%	-	-	-
GA aligned with STRidge	0.0143	0.0953	10.67%	-	-	-
R2 aligned with HuberRidge	0.0116	0.0847	9.56%	0.0011	0.0290	3.27%
GA aligned with HuberRidge	0.0144	0.0957	10.80%	0.0015	0.03416	3.85%

Table 2. Errors from *trainSTRidge* and *Iterative Huber Ridge* regression on experimental data aligned using R2 or Genetic alignment.

7. Discussions

The Genetic Alingment method we apply employs the lowess regression. However, depending of the parameters of the synthetic data, instead of smoothing the entropy, we should calculate a low or high enveloping curve. Further research on the entropy path given a patch size and the parameters we presume are behind the data...

The results we present lack confidence intervals. The p-values are not meaningful due to the large sample. It is possible to create confidence intervals from repetitive sampling. In the PDEs we estimate, we lack a positive growth term w. However, the mean of confidence interval for the growth term might be positive.

Another future venue in the area of sparse regression is the inclusions of initial conditions together with the PDE as a system of equations. Finally, residual nonlinearity could be captured using a kernel regression for the residuals.

8. Conclusions

Using synthetic data, misalignment increases the error and reduces the sparsity of the estimated PDE. Therefore, it is paramount to align the images before using regression.

In the quest for similarity we can lose dynamics. We have an identification problem because we cannot uniquely determine the parameters of a model based on observed data. We call this difficulty the Alignment Trap. The Genetic Alignment method avoids the alignment trap because it is based on entropy and not on similarity.

Iterative Huber Ridge is capable to handle outliers, this means that if we fail to align some patches, we can still work through. The PDEs we find with the Iterative Huber Ridge method on experimental data contain terms with derivatives, unlike the solutions found by trainSTRidge.

A. Appendix

In the algorithmic below, we present the pseudo code of the *Iterative Huber Ridge* used during this project:

Algorithm 1 Iterative Huber Ridge

Require: X: Input features, y: Target labels, α: Ridge regularization parameter, ϵ: Percentage of outliers supported by the model, $max\_iterations$: Maximum number of iterations, $learning\_rate$: Learning rate for updating weights, $tolerance$: tolerance value to compare the coefficients of the estimated equation.

Initialize weights w randomly or to zeros
Initialize iteration counter $iter = 0$
while $iter < max\_iterations$ **do**
  Compute predicted values: $y\_pred = X * w$
  Compute residuals: $residuals = y - y\_pred$
  for each data point i in X **do**
    if $|residuals[i]| \leq \epsilon$ **then**
      Compute Huber loss gradient for this data point: $huber\_grad = residuals[i] * X[i]$ {gradient of Huber loss function}
      δ is adjusted to achieve the ϵ percentage level of outliers
    else
      Compute Huber loss gradient with epsilon slope: $huber\_grad = \delta * \text{sign}(residuals[i]) * X[i]$ {gradient of Huber loss function}
      δ is adjusted to achieve the ϵ percentage level of outliers
    end if
    Compute Ridge regularization gradient: $ridge\_grad = -2 * \alpha * w$ {gradient of Ridge penalty}
    Compute total gradient for data point i: $total\_grad = huber\_grad + ridge\_grad$
    Update weights using gradient descent: $w = w - learning\_rate * total\_grad$
    for each w in the trained weights **do**
      if w tolerance **then**
        Keep the feature in the matrix X
      else
        Crop out the feature
      end if
    end for
  end for
  Increment iteration counter: $iter = iter + 1$
end while Trained weights w

In Fig.13, we plot the fitness of the best individual for each generation for the experimental data with a Fluence 0.20 and Delay 2. The fitness (MSE) declines at the beginning, later it stagnates until the 200th generation, where it declines until the 800th generation.

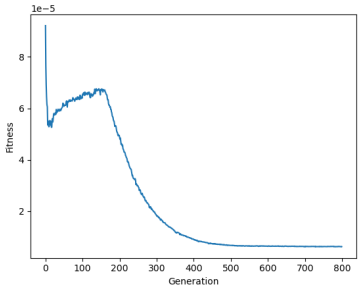


Figure 13. Evolution of the best individual fitness for each generations

In Fig.14, we present the entropy and the lowess approximation of misaligned data and the entropy of the best individual after 800 generations with the experimental data with a Fluence 0.20 and Delay 2.

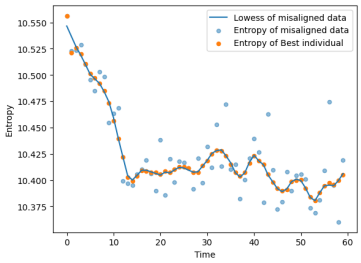


Figure 14. Entropy of the best individual, misaligned data and Lowess of misaligned data for the experimental data with a Fluence 0.20 and Delay 2

Below, in Fig15, we show a representation of the processes of the Iterative Huber Ridge, after each Huber ridge regression we compare the value of each weight to the tolerance value. If the coefficient is smaller, we take out the corresponding weight from the matrix of feature.

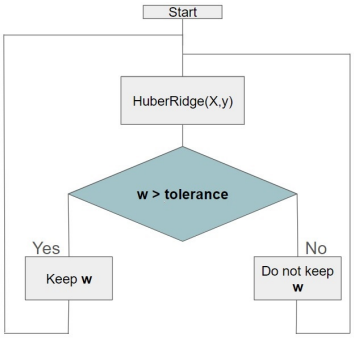


Figure 15. Iterative Huber Ridge

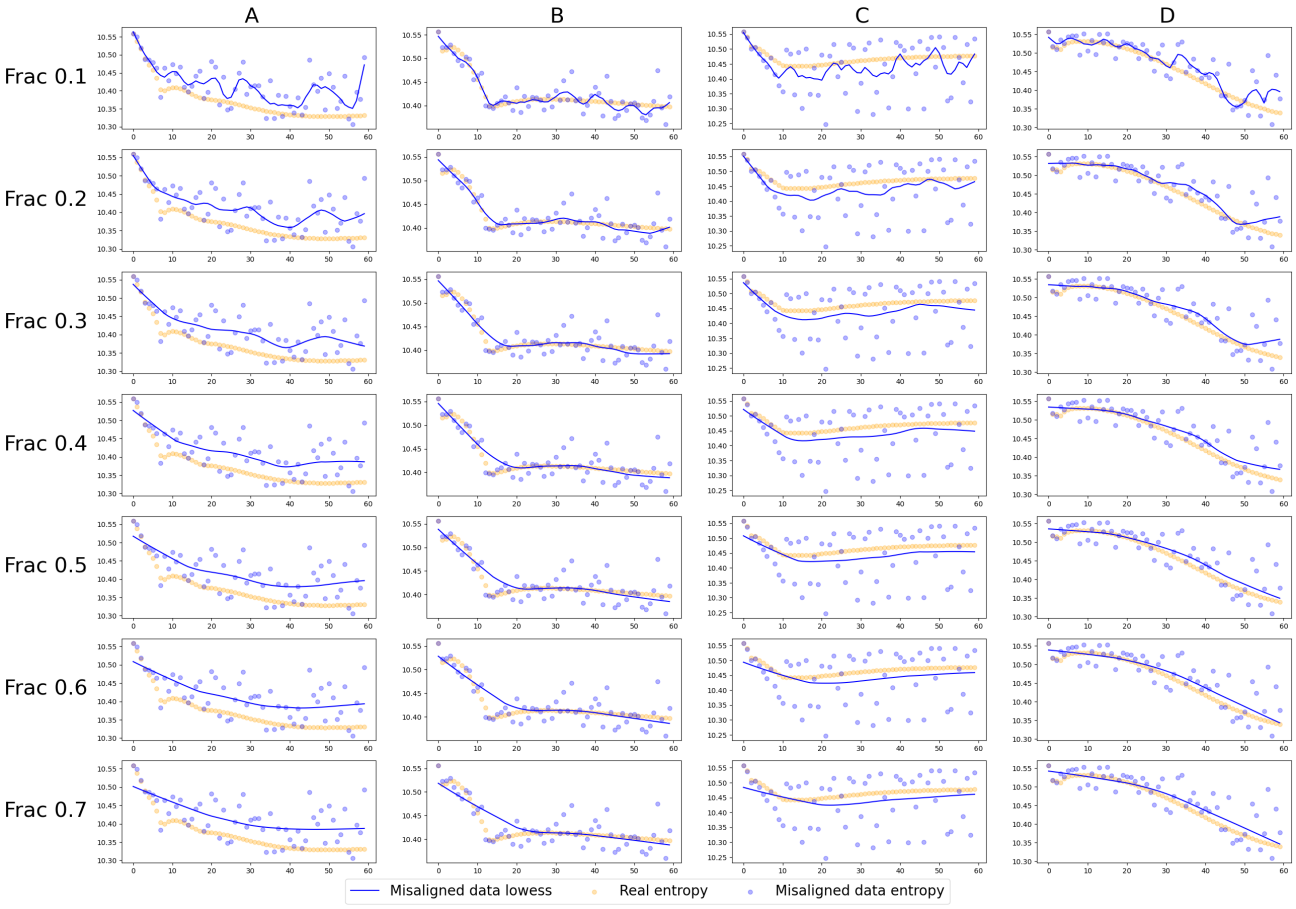


Figure 16. Synthetic date generated with different sets of parameters r and s: $A : (0.5, 2), B : (0.1, 2.2), C : (0, 2), D : (0.5, 1)$

References

Bertsimas, D. and Gurnee, W. Learning sparse nonlinear dynamics via mixed-integer optimization. *Nonlinear Dynamics*, 111(7):6585–6604, 2023.

Bramburger, J. J. and Kutz, J. N. Poincaré maps for multiscale physics discovery and nonlinear floquet theory. *Physica D: Nonlinear Phenomena*, 408:132479, 2020.

Chen, S., Brunel, N. J., Yang, X., and Cui, X. Learning interactions in reaction diffusion equations by neural networks. *Entropy*, 25(3):489, 2023.

Chen, Z., Liu, Y., and Sun, H. Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):6136, 2021.

Holland, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

Huber, P. J. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Jiang, Y.-X., Xiong, X., Zhang, S., Wang, J.-X., Li, J.-C., and Du, L. Modeling and prediction of the transmission dynamics of covid-19 based on the sindy-lm method. *Nonlinear Dynamics*, 105(3):2775–2794, 2021.

Kaheman, K., Kutz, J. N., and Brunton, S. L. Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A*, 476(2242):20200279, 2020.

Long, Z., Lu, Y., and Dong, B. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019.

Messenger, D. A. and Bortz, D. M. Weak sindy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021.

Nakhoul, A., Maurice, C., Agoyan, M., Rudenko, A., Garrelie, F., Pigeon, F., and Colombier, J.-P. Self-organization regimes induced by ultrafast laser on surfaces in the tens of nanometer scales. *Nanomaterials*, 11 (4):1020, 2021.

Pestourie, R., Mroueh, Y., Rackauckas, C., Das, P., and Johnson, S. G. Physics-enhanced deep surrogates for partial differential equations. *Nature Machine Intelligence*, pp. 1–8, 2023.

Rudy, S. H., Brunton, S. L., Proctor, J. L., and Kutz, J. N. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.