Analysis of Gas-Surface Interactions Using Relational Component Method and Mixture Density Networks

S. Poot [*] M. Sherif Osama Gamil[†]
1469886 1423630

Abstract

To predict the interaction between an argon gas molecule and a surface, we have introduced a new approach for calculating the outgoing velocities of the molecule based on its incoming velocities. This approach, named the Relational Component Method, is inspired by the study ”Deep Learning for Universal Linear Embedding of Nonlinear Dynamics.” [2]. It tries to decompose a data point into an array of weights, followed by a linear layer to a new array of weights, and then re-composes it into the output. By being able to decide what decomposition method and re-composition methods are used it is easier to understand what the weights and biases of the linear layer represent. Thus, making the entire method better explainable. It is compared with a feed-forward neural network to represent a baseline for the performance. The results indicate an element of randomness in the data which could not be overcome by the described regressive models. An alternative solution, which is to predict probability distributions rather than discreet values using a Mixture Density Network is proposed. The results aim to provide a baseline for the modeling task as well as an analysis of the data.

1 Introduction

Electronic chips are getting smaller and smaller. However, they also use more and more power which is then turned into heat. Thus, better heating systems need to be developed. One such heating system is by directly flowing gasses or liquids close to the silicon itself inside of the chips. To be able to design such a system, you need to be able to simulate the interactions on a very small scale. One such interaction is between the molecules of the gasses or liquids with a heated or cooled surface. Previous research already tried to compute the accommodation coefficients of the molecule wall interaction [7]. Instead of trying to find those coefficients, we tried to directly predict the molecule-wall interaction. Thus we have developed a new method which is a modification from a paper called ”Deep learning for universal linear embedding of nonlinear dynamics” [2]. We have changed it such that it should allow for better explainability.

The dataset is gathered from a simulation as explained in the paper ”Development of a scattering model for diatomic gas–solid surface interactions by an unsupervised machine learning approach” [7]. An example test setup is shown in figure 1

2 Methodology

2.1 Dataset

The dataset we used to design the relational component method is a dataset that records the collisions of gas molecules against an explicit wall. The explicit wall is a fine arrangement of other molecules where their movement is fixed except for the movement generated by the heat. The heat of each molecule in the wall is also fixed. The generated data for the dataset is as follows. When a molecule reaches a certain distance to the wall, then its velocities in the three dimensions are recorded. And after it again returned to the same distance from the wall after the collision the same 3 metrics are recorded. Thus each data point of the dataset has 3 metrics for the input and 3 for the output. We have used data of argon gas hitting an isothermal wall where there is no flow induced on the argon gas, and it has a fixed temperature.

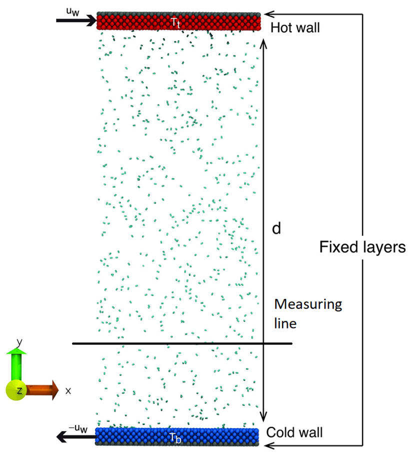

Figure 1: Experimental setup of the simulation. The measuring line is an example artificial boundary upon which the velocities of the molecules are measured

2.2 Relational component method (RCM)

In figure 4 you can see a diagram of the proposed method. It is modeled for the specific data we used.

The RCM is a new method derived from the paper called ”Deep learning for the universal linear embedding of nonlinear dynamics” [2]. In the paper it is described how using the Koopman operator and its eigen-functions, they can find coordinate transformations that make strongly nonlinear dynamics approximately linear. This theory is based upon the fact that it is impossible to linearly reconstruct the system state over the entire phase space with only a finite number of continuous observables when the nonlinear system admits multiple asymptotically stable equilibria. It becomes possible if you consider using discontinuous variables. This allows you to reconstruct the system in a weak sense [8].

The universal linear embedding of the nonlinear dynamics (ULEND) method has a single neural network that it uses as the encoder and decoder which approximates the Koopman eigenfunction. In between the encoder and decoder is a single linear layer. The relational component method follows the same structure. It has a decomposition method that is similar to the encoder where both try to transform the data in a new representation. Similarly, the re-composition method is similar to the decoder. It is however important to understand that the decomposer and re-composer are not necessarily trying to approximate the Koopman eigenfunctions.

[*]e-mail: s.j.poot@student.tuen.nl
[†]e-mail:m.mohamed.sherif.osama.gamil@student.tue.nl

The relational component method does not have a restriction to only using the Koopman operator to find a linear transformation of a non-linear function.

Although similar in concept, the RCM and ULEND have certain differences. A critical difference is that in the relation component method, we can choose the decomposition and re-composition methods as well as the base components. The idea of being able to decide the decomposition and re-composition methods is that you can better interpret the resulting input and output component weights and their relation. Consider the linear decomposition method 'fast Fourier transform'. If you would use this then the component weights are the amplitudes of the frequencies in the data. Then the relation between the input and output component weights is the relation between these amplitudes. Perhaps some frequencies of the input correspond to some frequencies of the output. Using a linear decomposition and re-composition method would not work because it makes the entire RCM linear and thus unable to capture the relations in a non-linear dataset.

Another difference is the choice of the base components. Base components can be seen as the order and filter of what you want to use for the component relations. Referring to the example of the fast-fourier-transform, the base components would be the frequencies of the dataset. In normal cases, you limit the amount of frequencies you use to approximate the original data. However, you can not only limit the frequencies used but also choose which frequencies to use.

2.3 Decomposition Methods

We tried multiple decomposition methods. Where each of them has their advantages and disadvantages. Some of the criteria we found were, whether they could also re-compose the decomposed weights. Whether they were implementable and if they were linear or non-linear.

2.3.1 Kernel Principal Component Analysis (KPCA)

The kernel principal component analysis is a kernelized version of the standard principle component analysis. Normal PCA tries to linearly separate the data. That is if you can draw a linear line between two different groups then it can be linearly separated. This however is not possible for nonlinear data. Thus KPCA first tries to re-project the data such that it becomes linearly separable by PCA. Then it runs normal PCA on the new projection. This re-projection is done using the kernel provided. The implementation used by this paper is from the scikit-learn library in the Python programming language [1]. Depending on what kernel it uses it can both be a linear or non-linear decomposition method. Two kernels have been used which are both non-linear. First, a polynomial kernel which computes the kernel as given in equation 1. The second kernel used is the radial basis function which is given in equation 2. These implementations are also given by the scikit-learn library in the python programming language.

The kernel principle component analysis together with the polynomial and the radial basis function kernel are the main decomposition methods used in the relational component method. This is because it has a standard easy-to-use implementation, it also has a built-in method for re-composing the decomposed data and is non-linear.

2.3.2 Empirical Mode Decomposition (EMD)

Empirical mode decomposition works differently from KPCA. It is a method that is a necessary step to reduce any given data into a collection of intrinsic mode functions (IMF) to which the Hilbert spectral analysis can be applied. It is done by finding the average between the maxima and minima in the signal. Then it subtracts it from the original signal and the process repeats until some criterion is reached. After this 'sifting' process you retain a signal that can be analyzed.

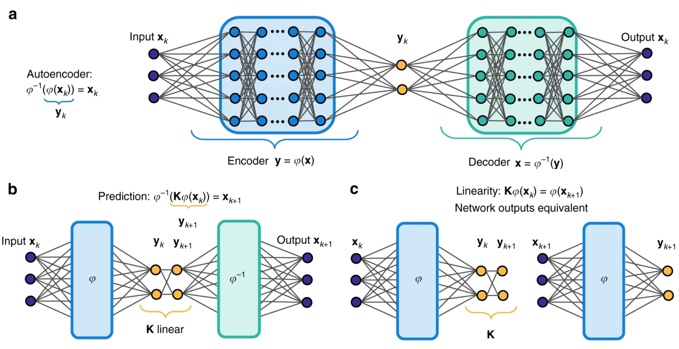
a
Input $\mathbf{x}_k$
Autoencoder: $\varphi^{-1}(\underbrace{\varphi(\mathbf{x}_k)}_{\mathbf{y}_k}) = \mathbf{x}_k$
$\mathbf{y}_k$
Output $\mathbf{x}_k$
Encoder $\mathbf{y} = \varphi(\mathbf{x})$
Decoder $\mathbf{x} = \varphi^{-1}(\mathbf{y})$
b
Prediction: $\varphi^{-1}(\underbrace{\mathbf{K}\varphi(\mathbf{x}_k)}_{\mathbf{y}_{k+1}}) = \mathbf{x}_{k+1}$
Input $\mathbf{x}_k$
φ
$\mathbf{y}_k$ $\mathbf{y}_{k+1}$
φ^{-1}
Output $\mathbf{x}_{k+1}$
K linear
c
Linearity: $\mathbf{K}\varphi(\mathbf{x}_k) = \varphi(\mathbf{x}_{k+1})$
Network outputs equivalent
$\mathbf{x}_k$
φ
$\mathbf{y}_k$ $\mathbf{y}_{k+1}$
K
$\mathbf{x}_{k+1}$
φ
$\mathbf{y}_{k+1}$

Figure 2: Proposed model from the "Deep learning for universal linear embeddings of nonlinear dynamics" paper.

2.3.3 Feed forward neural network (FFNN)

As one of the baseline comparisons for the performance of the RCM a standard feed-forward neural network is used. It starts with a dense layer with 128 nodes with "ReLu" as an activation function. The data is then normalized and given to another dense layer with 128 nodes and "ReLu" as an activation function. Then a dropout layer with a 50% chance of dropout. Followed by another dense layer of 256 nodes with a "ReLu" activation function. Then the data is again normalized and put through a dense layer of 256 nodes with a "ReLu" activation function. Then another dropout of 50% chance of dropout and finally a dense layer of 3 nodes for the $v_{x,out}$, $v_{y,out}$ and $v_{z,out}$

2.3.4 Other methods

A few other methods have been attempted. First, multivariate decomposition is a method that tries to separate groups using regression such that some distance is maximized between the groups. What makes it a non-linear decomposition is by choosing a non-linear distance metric. A few have been investigated in the past such as the logit function, the probit function, Poisson, negative binomial and the complementary log-log [4]. The paper we found however only provides a decomposition implementation in Matlab. We could not find a suitable re-composition with it and thus we did not investigate further into the performance.

Then comes the nonlinear mode decomposition. This decomposition tries to identify the most important harmonics first. Then it reconstructs a new mode given these harmonics and removes the mode from the original signal. After which it repeats until a certain criterion is reached [5].

And lastly, data-driven participation factors for nonlinear systems based on Koopman mode decomposition. This method relies on identifying the participation factors in the data. These factors describe how important each of the Koopman modes are [6].

The main problem with these three methods is that they do not have a means of reconstructing the original data. This still allows us to train the RCM as described in section 2.2. But in the end, we will not be able to use it to predict new data. However, each had their own problems such as only being able to be used for 1 dimensional data for the nonlinear mode decomposition. Thus the decision has been made to not further use these three methods for the investigation of the RCM.

$$k(x,y) = (\gamma x^\top y + c_0)^d \tag{1}$$

$$k(x,y) = \exp(-\gamma \|x-y\|^2) \tag{2}$$

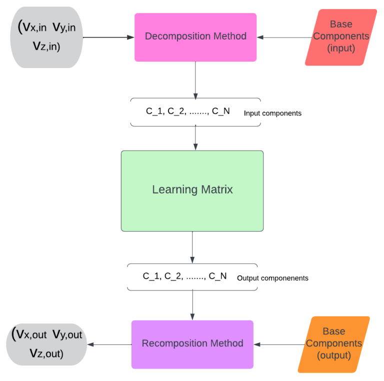

Figure 3: Diagram of the relational component method.

3 Experiment and Results - RCM and FFNN

We run multiple experiments, implementing both RCM and FFNN for the task of predicting exit velocities of Argon gas given the input velocities. The data was randomly shuffled and split with 80% for training and 20% for testing. For all models, the training was run until convergence. Root mean squared error (RMSE) was used as the models' loss function and main evaluation metric. See Table 1 for an overview of the results.

Table 1: Quantitative comparison of exiting velocity prediction

Method	Final RMSE	Details
RCM-EMD	2.326	IMF = 10
RCM-KPCA	2.529	rbf/20 components
RCM-KPCA	2.835	poly/20 components
RCM-KPCA	2.810	poly/200 components
FFNN	2.2264	118147 parameters

3.1 Relational component method

As you can see in table 1 the final testing loss of the RCM using the KPCA with the polynomial kernel is 2.810. These metrics are both given by the root mean squared error. The entire loss curve over time can be seen in figure 4. The figure shows that the RCM method can properly converge. Suggesting its capability to learn from the data and thus also provide meaningful results in the linear layer as per design. However, from the table, we can also see that the EMD decomposition gives better results. Indicating that some decomposition methods give better results, at least for the data used. However, we can also see that the RCM converges to around 2.xx and is unable to get better losses. This prompted us to investigate further into how well the RCM works, by looking at how well the linear layer can capture the relation between the input component weights and the output component weights.

Due to the seeming bottleneck of the 2.3 loss that the RCM gives, we decided to investigate how well the decomposition methods can linearize the system. Of course, continuing linearizing a system is not possible as previously stated but the decomposition method should thus use discontinuous observations and locally linearize the system, which should be possible as previously stated. The results are given in the figures 5 and 6. In these figures, the first three-component weights relations are displayed. That is, the linear relation between the first component of the input and the first component of the output is in the leftmost figure, and from left to right you get the second component of the input with the second component of the output and so on.

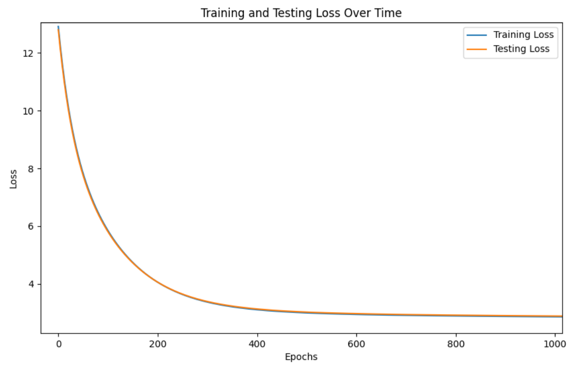

Figure 4: This figure shows the loss of the RCM using the KPCA with the polynomial kernel.

If the KPCA decomposition method was able to convert the non-linear dataset into linear components then these figures should display a linear relationship. As seen in the figure, this is not the case. This can be because of several reasons. Either the data can not be captured properly by the decomposition methods, these decomposition methods are not the right methods for locally linearizing the system, or we have not fixed the base components properly. Consider that sometimes one component can be more or less important when looking at different data. In the implementation, the most important component always goes first. But this component might thus change per data point which suggests that the relation the linear layer tries to capture is constantly changing.

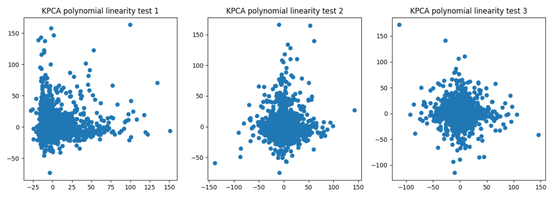

Figure 5: This plot shows the relation between the component input weights (x-axis) and the output component weights (y-axis) for the first 3 component weights for the KPCA decomposition method with the polynomial kernel

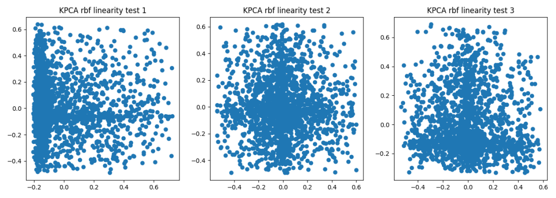

Figure 6: This plot shows the relation between the component input weights (x-axis) and the output component weights (y-axis) for the first 3 component weights for the KPCA decomposition method with the radial basis function kernel

3.2 Feed Forward Neural Network (FFNN)

From the table 1 we can see that the FFNN can get a testing loss of 2.2264. This loss is close the the loss of the RCM-EMD. This suggests that they both hit a similar bottleneck. Since both models try to output individual data points from individual data points in the input. It is suspected that the data contains one or more hidden random variables which such models can not capture. On the other hand, we know that the wall that is used for the simulation is not a specular wall that reflects the gas molecules the same way every time. Instead, the wall is modeled in an explicit matter, such that the gas molecules' interactions with the surface are modeled as explicit collisions with the wall molecules. However, the wall molecules are not uniform or level throughout the wall (rough surface), thus adding to the random element. This entails that there may be additional information that the data cannot fully capture by observing the gas molecules on their own. Hence, correlation in the data can not be completely captured by the proposed RCM or the standard FFNN.

4 Mixture Density Network (MDN)

As mentioned previously, due to the nature of the problem and the high degree of variability, randomness, and uncertainty in the data, a regressive model that outputs a single value for the output velocity will not suffice. Instead, a probability distribution as a prediction serves as a more suitable solution. On the other hand, the negative log-likelihood to compare the predicted distribution to the ground-truth data becomes the preferred evaluation.

We consider Mixture Density Networks (MDN) as they are a popular candidate for this type of model and have been previously used to model gas interactions [9]. An MDN typically consists of three main components: a neural network that processes the input data, a set of output layers that produce the parameters of the mixture components, and a mixture model that combines these components. The neural network processes the input data to generate the parameters of the mixture distributions, such as the means, variances, and mixing coefficients. These parameters define a probability density function, which can then be used to sample from the predicted distribution or to calculate the likelihood of observed data [3].

We proceed to implement and test the model to set a baseline for the task of predicting velocity distributions.

4.1 Experiments and Results - MDN

We design an MDN with 84 parameters and an output shape representing 3 distributions for the output velocity across the x, y, and z dimensions. The data was shuffled and scaled between -1 and 1 before split into a train-test ratio of 80/20. The model was trained with an early stopping mechanism (patience = 10). The model's training criteria were to minimize the negative log-likelihood of its output.

Figure 7 shows the train and test negative-log-likelihood loss over the epochs. The final loss was (-0.133).

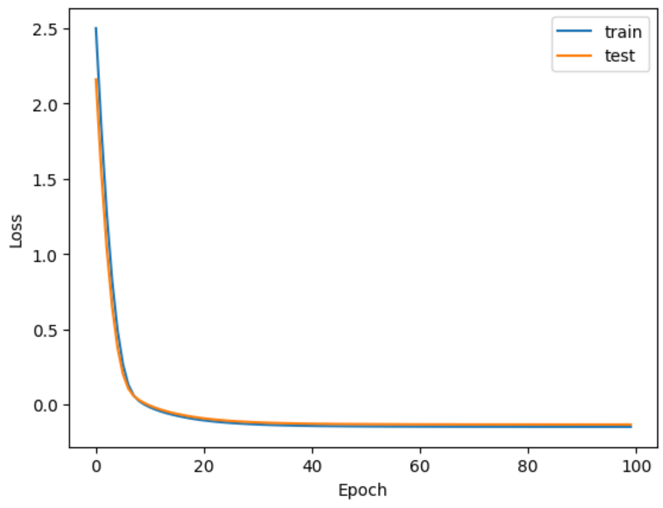

Figure 7: Training and test loss over time - MDN

5 Conclusion

From the results, we can conclude various things. The first is that the data can not be captured by any of the tools used in this paper. Whether we use the RCM or we use a FFNN. All show similar losses when using the root mean square error loss. This would indicate that all tools arrive at the same problem and thus it would be better to investigate if something can be done about the data rather than the methods applied. Considering that the wall used is a simulated wall with a fixed molecule grid and temperature. It might be the case that merely taking the velocities of the gaseous molecules into account might not be enough, as the position of the collision on the wall is an important factor.

The second is that the RCM needs a suitable decomposition and re-composition method that can convert a nonlinear dataset to a linearized representation. We are unable to verify if this is the case with the attempted methods due to the dataset. Thus, further study will be necessary to understand if any of the attempted methods can do so, which will in turn decide whether an RCM is a viable method for finding, training, and explaining important relations in a non-linear dataset. An important note is that decomposition methods need to be accompanied by straightforward mechanisms to recompose the data, which is not the case for all of them.

Given the findings and results of the RCM and FFNN implementation, it was decided that the problem description was to shift into predicting distributions rather than discreet values. We attempted the Mixture Density Network to offer a baseline, with respect to negative-log-likelihood, for this solution. Further analysis, such as using these distributions to extract accommodation coefficients, is necessary to properly evaluate the relevance of the results.

References

[1] Kpca. July 2024. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html.

[2] S. L. B. Bethany Lusch, Nathan J. Kutz. Deep learning for universal linear embeddings of nonlinear dynamics, November 2018. doi: 10.1038/s41467-018-07210-0

[3] C. M. Bishop. Mixture density networks. 1994.

[4] M.-S. Y. Danial A. Powers, Hirotoshi Yoshioka. mvdcmp: Multivariate decomposition for nonlinear response models. *The Stata Journal*, 2011.

[5] A. S. Dmytro Iatsenko, Peter V. E. McClintock. Nonlinear mode decomposition: A noise-robust, adaptive decomposition method. *Journals.aps.org*, 2015. doi: 10.1103/PhysRevE.92.032916

[6] L. M. Marcos Netto, Yoshihiko Susuki. Data-driven participation factors for nonlinear systems based on koopman mode decomposition. *arxiv.org*.

[7] S. Mohammad Nejad, S. Nedea, A. Frijns, and D. Smeulders. Development of a scattering model for diatomic gas–solid surface interactions by an unsupervised machine learning approach. *Physics of Fluids*, 34(11):117122, 11 2022. doi: 10.1063/5.0110117

[8] K. D. Shaowu Pan. On the lifting and reconstruction of nonlinear systems with multiple invariant sets. *Nonlinear Dynamics*, 112, 2024. doi: 10.1007/s11071-024-09581-0

[9] A. L. B. Vollebregt. A machine learning-based energy exchange model for rarefied gas flow in dsmc. Master's thesis, Eindhoven University of Technology, November 2022.