Comparison of Data-Driven Approaches to Modeling Complex Behavior of 2D Liquid Simulator

Dmitry Kovalev[1], Dmitry Khliustov[2], Sergey Safonov[1]

[1] Aramco Research Center, Moscow, Aramco Innovations LLC, Russia
[2] Lomonosov Moscow State University, Russia

dmitry.kovalev@aramcoinnovations.com
hlustov.d@gmail.com
sergey.safonov@aramcoinnovations.com

Abstract. Modeling complex system dynamics traditionally is implemented with the use of differential equations, which requires hand-crafted work of a qualified expert and significant amount of time. The advent of data-driven approaches allows to overcome these difficulties and substitute traditional models with models built in automated way directly from observations. This paper compares several data-driven approaches to modeling 2D liquid simulator. Dataset is generated from it for both training and testing with fixed simulator parameters. Local and global types of models are evaluated with metrics, describing different aspects of liquid behavior (spatial, spatio-temporal and worst-case settings). Other metrics introduced allow to capture differences not only in distances, but also in distributions, which is more natural for human perception and enables to quantitively compare similar pictures. From the model evaluation, it is inferred that the use of decomposition improves overall accuracy and the trajectories figures, though at the same time model generalizability decreases. On the other hand, utilizing locality leads to more generalizable models at the cost of accuracy. Model training and inference times are provided and main directions for further research are outlined.

Keywords: 2D liquid simulation, dynamic mode decomposition, neural ODE, graph neural networks.

1 Introduction

The problem of modeling complex systems dynamics has been of great interest to scientists and engineers for a long time. Classical approach to this task involves using calculus and partial differential equations. The main advantage of it is that equations can be treated from analytical point of view, enabling researcher to describe behavior of solution completely and either apply the model to practical tasks or develop another one based on the original model. However, most of equations obtained are rather complex, and their properties can only be described using highly non-trivial methods of different branches of mathematics. Developing differential equations requires

hand-crafted work of a qualified expert and requires significant effort and amount of time.

Another approach to modeling complex systems has become possible with data deluge coming from sensors and simulators. Bottom-up data-driven approach requires little or no expert knowledge and is aimed at extracting the dynamics of complex system, usually in a form of system of equations. This approach has multiple benefits, including the huge decrease in computational time for modeling and no prior domain expertise [1]. Data-driven methods allow orders of magnitude faster inference without losing accuracy in several domains [2].

Prominent example of complex system are liquid simulators. There exist two main simulator types based on Eulerian grids and Lagrangian particles. Simulators can produce both three-dimensional and two-dimensional datasets. In [3] a two-dimensional liquid simulator is implemented with JavaScript and Python. Simulator is based on smoothed particle hydrodynamics, particularly double density relaxation. First, the velocity of each particle is updated according to Newton's Second Law: accelerations are computed by summing up forces which impact the particle. The second step consists of calculating a kind of local density around each particle: the coordinates of its nearest neighbors are convolved with a kernel and summed. If the resulting quantity in a given point is higher than average, extra impulses are added to neighbors, thus lowering the density in current region. Similarly, if the quantity turns out to be lower than average, the simulator tends to increase it. After these two steps particle coordinates are updated by adding infinitesimal movement vectors, that is the product of velocity and given length of time step. Simulator enables to tune multiple liquid parameters, such as viscosity, gravity, density, and stiffness. There can be multiple emitters with adjustable frequency, angle and angle velocity, emission strength. Particle coordinates (up to 1500 particles) for consecutive time steps are extracted from the simulator. It is also worth noting that that simulator has a tunable noise parameter, that is, one can add stochasticity to particle movement, thus obtaining more complex dynamic patterns. Absolute values of all the particle coordinates do not exceed 150.

To achieve generalizability initial and boundary conditions were changed from simulation to simulation, so that dataset consists of multiple runs with different boundary and initial conditions. A configuration with two emitters is chosen since it allows to observe multiple particles collisions and nonlinearity in change of their directions while not making the system too complicated to simulate (see Fig. 1). The four parameters of the liquid are fixed, their values chosen in a way that allows one to obtain complex patterns in liquid. Emitters' coordinates x_i, y_i, angle θ_i and initial particle velocity λ_i were randomly generated, and configurations with the most complicated particle patterns have been chosen manually. The total of 100 tuples of conditions were formed, and each simulation computed for total length of 15 seconds. After training the model on one simulation, its performance has been measured on all the others.

There are three possible ways one can state the problem of behavior simulation. The first one, which is referred to as basic, is one-step prediction. That is, one aims to reconstruct particle coordinates in moment *T+1* using data for moments *1, …, T*. After

this, in order to predict coordinates in *T+2* moment, true data samples for *1*, *..., T+1* are used. Usually, one does not need all the past data, and correct prediction can be made from timesteps *T-p*, *..., T*. Thus, for each time step model works only with data obtained from simulator and is not using its own predictions.

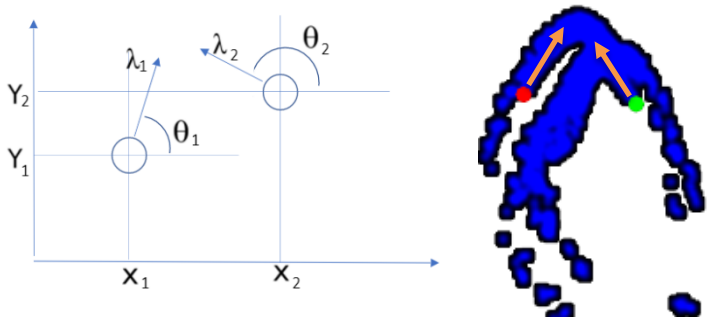

Fig. 1. Configuration parameters (on the left) and experiment example with two emitters (on the right)

The second problem statement, namely rollout prediction, is formulated as follows. The model tries to recover system dynamics for timesteps *T+1*, *... T+l* using data samples collected for timesteps *1*, *..., T*. This is a much more complex task: even small disturbances of predicted coordinates may lead to exponentially fast deviations from true trajectories even after few steps. The possible solution proposed for this problem consists of adding noise to the training samples, thus forcing the model to produce more stable predictions, which are not as sensitive to small disturbances. An example for these two ways of stating the problem is depicted in Fig. 2. Although usually the one-step prediction model produces the trajectory close to the true one, the same model applied in rollout way leads to large deviation from the target.

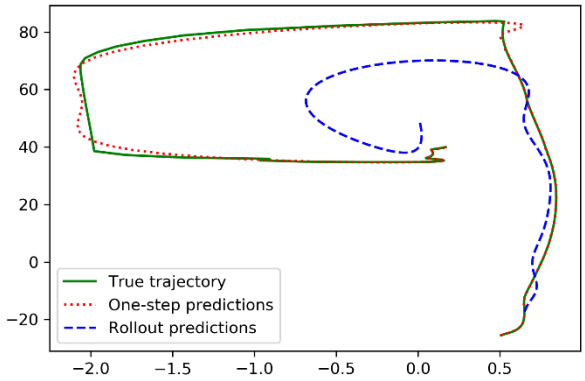

Fig. 2. Single particle trajectories computed with for the first and second problem statements

Finally, the third class of problems one can state is the inverse modeling. Possible problems statements for inverse modeling involve reconstruction of the emitter con-

figuration and strength from data available. However, this problem cannot be solved in general, as one can construct two configurations that would result in very similar particle patterns after few timesteps. Therefore, the problem is restricted to the task of predicting particle coordinates just for one step backwards: from data for $T+1, ..., T+l$ the aim is to obtain coordinates for the moment T.

This paper is aimed at working with basic problem for predicting one step and does not tackle rollout and inverse problem statements. The rest of the paper is organized as follows. In section 2 related works are presented. In section 3 a brief introduction into theoretical aspect of approaches to simulation and their combinations is provided. In section 4 accuracy metrics are discussed. Section 5 presents simulation results and their evaluation, as well as their metrics dependency on data and hyperparameters. In the last section conclusions are presented and way forward is outlined.

2 Related Works

Data-driven approaches are widely applied in physics, climatology, epidemiology and other highly computational branches of science. All the methods chosen for this paper either belong to a class of most modern, or have recently been rediscovered, and thus are in the focus of research across many disciplines. The main directions of studies are model accuracy, computational speed and generalizability. Various results are obtained by combining different models, thus incorporating benefits and reducing drawbacks of individual approaches. Another direction of research is the increase in interpretability of models, development of data-driven methods based on general principles of complex systems behavior.

In [4] the problem of simulating liquid behavior is studied. In this work Proper Orthogonal Decomposition (POD) is used for extracting temporal patterns and Neural Ordinary Differential Equations (NODE) are applied to predicting spectral coefficients. The performance of NODE is compared to LSTM, and superior accuracy of the method proposed by authors is detected. However, the border and initial conditions of the system studied lead to emergence of stationary flow and no generalization attempts have been made.

In [5] NODE is applied to the problem of forecasting turbulence in liquid. The performance of data-driven model I compared to those of direct numerical simulation based on Navier-Stokes equations. It is demonstrated that different kinds of transient phenomena are captured accurately and forecast for a long time period is possible. However, the generalization properties of the method proposed are not studied, since the system of interest is fixed, and parameters of simulation varied only slightly across different runs.

In [6] Graph Neural Networks (GNN) are applied to the problem of modeling liquid behavior. The encoder-processer-decoder architecture is implemented, and special mechanism of message-passing between different parts of the system is used. Complex mixtures of materials are studied, and results are measured by MSE and distributional metrics. It is stated by the authors that performance of GNN is close to that of

classical methods (such as convolutional), but better generalizability is achieved in case of different materials interacting.

In [7] decomposition method is combined with predictive model. Ensemble Empirical Mode Decomposition (EEMD) is used for extracting spectral patterns in data, and Time Convolutional Neural Networks (TCN) are applied to the resulting amplitudes. The performance is measured by Root Mean Squared Error (RMSE) and Pearson Correlation Coefficient (PCC), and the results are compared to those obtained by TCN without decomposition, Long Short-Term Memory (LSTM) and LSTM with EEMD. It is outlined that the combined model performs better in cases with low-variance data but fails in accounting for high-amplitude local disturbances, which usually can be found in real-world oscillatory data.

In [8] various applications of GNN to particle physics are studied. It is outlined by authors architectural choice is task-specific, and different mechanisms of local subsystem update are described. Some methods, such as attention mechanism, are formulated in terms of GNN, which leads to higher physical interpretability of models. The mechanism used in our paper is much simpler, since the system of interest does not contain non-homogeneous parts. However, future research may be focused on the effect of incorporating other protocols of interaction between individual subsystems.

In [9] CNN are used for the problem of flow-field decomposition. The neural network-based approach is compared to POD, which may be considered classic in the field of hydrodynamics. The main conclusion stated by authors is that CNN-based decomposition models can capture complex patterns and transient phenomena, which are usually misrepresented by POD. The main reason DMD has been chosen for similar purposes in our paper is that it possesses an ability to accurately extract complicated dynamical patterns in the data, same as CNN, but is usually much faster to compute.

In [10] NODE model is applied to the study of ecological system, consisting of hares and lynxes. It is demonstrated that one can describe time dynamics of the population with high accuracy, capturing seasonal and cyclic phenomena. However, it is noted by authors that NODE is computationally expensive in this case because of non-linear character of equations and strong coupling between variables. This drawback might have been crucial for the problem studied in our paper because of the high number of particles with coupled dynamics. However, the simplicity of individual behavior and locality of interaction enables one to reduce the number of model parameters. Decomposition methods simplify the data even further, enabling one to reduce computational time drastically.

The contribution of this paper is the following. First, a complex multi-agent system is successfully simulated by means of data-driven methods. Second, these methods are compared in precision (by means of different metrics), generalization capabilities and by training and working time. Third, two novel method combinations (namely NODE with DMD and CNN with DMD) are introduced, and their properties studied. Detailed explanation of results obtained is provided and main benefits and drawbacks of all the methods is provided, together with physical and geometrical interpretation of performance nuances.

3 Data-driven Approaches to Modeling Complex Behavior of 2D Liquid Simulator

The system dynamics can be predicted globally for all particles at once and locally for selected particles. Five data-driven approaches to the problem of modeling liquid behavior are utilized. Three of them are directly taken from the related works, while the last two are proposed by the authors. Model comparison is summarized in Table 1. Models are described in detail below.

Table 1. Comparison of surveyed approaches.

Model	**Type**	**Advantages**	**Disadvantages**
Dynamic Mode Decomposition	Global	Simple and Fast Interpretable	Accurate only for a small class of systems Non-applicable for highly non-linear systems
Graphical Neural Networks	Local	Based on physical principles Generalizable Requires less timesteps than other methods	Requires additional step of extracting graphs from data Graphs are recomputed at each step Requires system to be homogenous
Neural Ordinary Differential Equations (NODE)	Local	Strong theoretical foundations Expressive power	Theoretical limits Computationally intensive
NODE/CNN with DMD	Local/ Global	Faster compared to similar models without decomposition Small number of hyperparameters	Not generalizable

3.1 Dynamic Mode Decomposition

The first one, Dynamic Mode Decomposition (DMD) [1] stems from the apparatus of linear algebra. Each time snapshot is flattened into a vector, which is composed of coordinates of each particle in corresponding moment (see Equation 1). The idea behind DMD is developed from the Koopman theory [11]. First one may note that any dynamical system can be viewed as a linear one in an appropriate basis. Each function studied is an element of Hilbert space, therefore the manifold of all the solutions is some subspace. Hence for a wide class of systems one can represent the governing differential operator as a linear one with domain in the original space. Although the choice of the suitable space is a complex problem, sometimes by combination of feature engineering and decomposition appropriate description of the system in terms of Equation 1 can be obtained.

$$X = \begin{pmatrix} | & | & & | \\ x_0 & x_1 & \dots & x_{n-1} \\ | & | & & | \end{pmatrix}, Y = \begin{pmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_n \\ | & | & & | \end{pmatrix} \quad (1)$$

$$AX = Y$$

The DMD searches for a linear shift operator *A*. That is possible since as the number of time steps grows large enough and most important physical features are captured, vectors from Equation 1 become linearly dependent. Decomposition may be viewed as a variation of singular value decomposition (SVD) problem. Similarly, SVD is calculated for *A*, and small singular values are omitted to keep the size of the operator small. Singular values large enough are called modes, and the dynamics thus described is much easier to compute than the original one but is usually not much less accurate. The main disadvantage of this approach is that not any system can be approximated by linear operator in finite spaces. However, in most practical cases DMD turns out to be extremely useful.

3.2 Graph Neural Networks

The second approach, Graph Neural Networks (GNN), utilizes geometrical methods [6] and is based on assumption that system dynamics is determined by local behavior. In case of liquid to describe dynamics, one needs to obtain description of individual particles and their interactions, which may be thought of in terms of directed graph (see the example in Fig. 3) with predefined number of neighbors *k*.

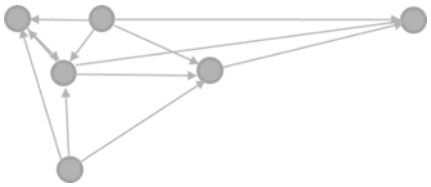

Fig. 3. Liquid graph for particle number $n = 6$ and neighbor number $k = 3$

One can notice that all the particles in the liquid have the same behavior under same circumstances (homogeneity), and that the dynamics of each particle depends only on the characteristics of those which are located in some neighborhood of it, typically small in comparison with the size of the whole liquid space [12]. Hence only one update function is needed, which would be applied iteratively to each of local clusters of particles. One may write the following equation in order to show this process:

$$S^i{}_{t+1} = f\left(S_t^i, \{S_t^j\}\right), \ for\ j \in Neighborhood(i) \quad (2)$$

where S_t^i denotes the state of i^{th} particle in a time moment *t*. In this study only particle coordinates were included into state vector, however in case of liquid mixtures more complex vectors (e.g., adding liquid properties) would be used. It is important to

note that the function f does not depend on *i*, which comes from assumption of liquid homogeneity.

The main disadvantage of this approach is that it is not clear without experiments how to choose the number of neighbors *k* and what architecture of individual network to use. However, it is easy to obtain huge training dataset utilizing liquid homogeneity (we can get *N* training samples out of every snapshot, where *N* is the number of particles). Utilizing these two facts, graphs are formed for each particle and its *k* nearest neighbors, which would predict behavior of only one particle conditional on behavior of its *k* nearest neighbors for some *k*. Later, to obtain the exact value of *k* when liquid is simulated globally as a whole, *k* nearest neighbors of each particle are computed for *k* in range from *1* to the number of particles, local dynamics is simulated from data and next the global behavior of the liquid is reconstructed. The optimal value is chosen based on reconstruction error.

3.3 Neural Ordinary Differential Equations

The idea behind the Neural Ordinary Differential Equations (NODE) stems from the fact that the equation governing the dynamics of residual neural network looks much like the one describing Euler's method of solving ODE:

$$z_{t+1} = z_t + h * f(z_t, \theta, t) \tag{3}$$

where z_t denotes the output of t'th layer, f is an activation function, depending on some parameters. This recurrent equation leads to the differential one:

$$z'(t) = f(z(t), \theta, t) \tag{4}$$

Therefore, instead of teaching networks one can solve and optimize ODEs with respect to parameter. The authors of [13], who were the first to introduce this approach, suggest using the adjoint sensitivity method. While the residual networks have been constructed as those having adaptive depth, NODE can be thought of as having infinite depth. Therefore, the expressive power of this class of models is rather high.

However, few theoretical limitations exist. The main drawback of ODE of any kind is that under assumption of smoothness of right-hand side *f* (which is typically the case) the trajectories of solutions cannot intersect. Hence, many functions possessing mixing property are not approximated by them. The solution to this problem consists of adding extra dimensions to data, thus enabling trajectories to move "one below the other". However, the choice of optimal augmentation dimension is a complex problem itself. In this paper no augmentation is used, since particle dynamics is smooth and there are no mixing effects as a result.

3.4 Combinations of Decomposition and Continuous Models

Related works suggest that very high dimensional systems can be reduced to the latent space of much smaller size without severe loss of information. Authors propose a novel approach which combines both DMD and NODE and is called NODE with

Decomposition (NODEwD). Namely, data have been decomposed by means of DMD, and spectral coefficients were predicted with NODE. Thus, under assumption that global patterns of particle organization do not change over time (which is typically the case for stationary dynamics) the evolution of the system has been studied.

There exist several attempts to combine decomposition with predictive models, however, not many of them devoted to the problem of modelling liquid behavior. The authors of the paper [7] use Ensemble Empirical Mode Decomposition for extracting patterns in data and make predictions by means of Temporal Convolutional Networks. However, the paper deals with the real-world data, thus the noise in the measurement must be accounted for. In case of noiseless data, DMD may provide more accurate and stable results, since longer time dependencies are extracted by this algorithm.

This approach resulted in dynamics very similar to the true one. Since the configuration of emitters did not change over time, the model was able to reconstruct liquid behavior from global patterns, represented by vectors of DMD.

In order to study whether the success of this approach was due to NODE properties or the decomposition, the comparison has been drawn between performance of this model and the similar one but using another class of neural networks – convolutional neural networks [14] (CNN) for predicting spectral coefficients and combined with decomposition is called in this paper CNNwD. The results clearly indicated that performance metrics of this modified model were not much different from the NODE approach.

The main drawback of this approach is its low generalization abilities. If the initial conditions of the original simulator are significantly changed, patterns of particle organization are changed also, hence making the learned model nearly useless for the task of predicting liquid behavior in this changed environment. That drawback can be overcome with increasing the dataset with other configurations which also results in training time increase.

4 Metrics

Main metrics used in this paper are from the [6]. The basis of all the studies is mean squared error (MSE), calculated for different data clusters. Possible approaches to the problem of evaluating model performance are summarized below in Table 2.

The first metric, MSE, is widely applied for data-driven models. Two other types of MSE, row-wise and uniform, are specific to 2D time-series data. They are regular MSEs, calculated for each timestep and for each particle in each timestep respectively. They are used for more detailed study of particle distribution and are depicted by means of histograms. The supremum metric represents the maximal error the model makes while predicting particle coordinates in each moment.

Two metrics, Optimal Transport (OT) [15] and Maximum Mean Discrepancy (MMD) [16], are widely applied in statistics. They both concern probability distributions and measure the distance between them. If one treats probability density functions as mass distributions, OT metric would represent the minimal value one must pay in order to re-distribute all the mass of first probability density function and ob-

tain the second one. Precise mathematical definitions are provided in Table 2. OT allows to treat a case where, for example, the model has swapped two particles, which would not alter the picture but will lead to increase in MSE. MMD is another probabilistic metric, and it is closely connected to the theory of weak convergence [17]. Its properties are similar to OT. It is important to note that in this work distances between empirical distribution functions are measured, that is the mass of any subset of the plane equals the portion of particles contained in it.

The last metric chosen is the Hausdorff distance, a common topological similarity measure. It allows one to calculate the distance between two compact subsets of a space, which is exactly the case with particle simulation. This paper does not focus the MMD and Hausdorff distance, since they are quite similar in properties to the others studied.

Table 2. Proposed metrics

Metric	**Type**	**Formula**	**Pros**	**Cons**
MSE	Number	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i}^2)$	Differentiable Widely applied	Does not capture all the aspects of liquid behavior
Optimal transport	Number/ Histogram	$\inf_{\gamma} \int c(x,y)d\gamma(x,y)$ where c is a cost function	Captures the effect of particle swapping	Not easy to compute Equivalent to MSE between true positions and some permutation of predicted coordinates, thus all the cons of MSE
Maximum mean discrepancy (MMD)	Number/ Histogram	$\sup_{f \in F} E_p(f(x)) - E_q(f(y))$ F is some class functions	Captures the effect of swapping	Exceeds supremum metric, thus quite high
Row-wise MSE	Histogram	MSE for each row	More accurate than simple MSE	Sensible only for relatively good and smooth predictions
Uniform MSE	Histogram	MSE for each particle in each moment	Captures global model precision	Difficult to compare two models
Supremum	Histogram	Maximal error in particle prediction for a given timestep	Allows to study worst-case performance	Always very large

5 Application and Results

5.1 Restoring Simulator Dynamics

In Fig. 4 whole system dynamics is depicted for multiple models. Trajectory of each particle is depicted with different color. Particles are emitted in two places and are shot in different directions.

One can notice that the DMD result is the worst – all the complex patterns are lost, and only main directions of movement can be seen. However, this model may be useful for brief description of dynamics when details do not matter much. It is worth mentioning that the rank of DMD was chosen to be 20 – lower ranks resulted in a worse picture, and higher did not bring much better precision with them.

NODE and GNN results look quite similar to each other. One may notice that trajectories of individual particles, fluctuating along the path, do not let researcher to see the details in the picture. However, as metrics indicate, these predictions are rather precise.

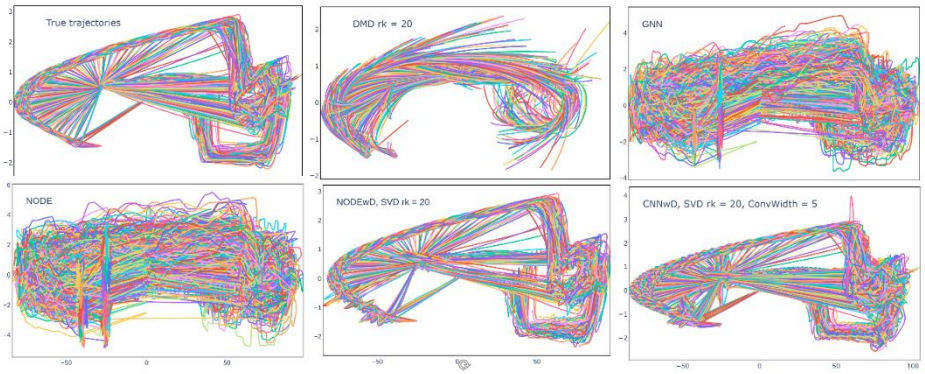

Fig. 4. Particle trajectories generated by simulator and by different models

The results computed by NODEwD and CNNwD also do not differ much. Although fluctuations in regions with complex dynamics can be noticed, they do not affect the picture much, thus making predictions close to the true data.

In order to demonstrate the idea behind distributional metrics in Fig 5. the shift from source to target samples is depicted for the best (NODEwD) and worst (DMD) model. The first row provides an example of calculation of OT metric for DMD and NODEwD for a chosen time moment. A calculation of OT metric for DMD and NODEwD for a chosen time moment. Blue crosses represent true positions of particles, while red ones denote results of model predictions. In order to compute the cost of empirical distribution deformation, for each particle its neighbor is searched for such that the total distance (sum of lengths of blue lines) is minimized. The total distance of shifts for the DMD model significantly exceeds respective total distance for the NODE with decomposition, which coincides with other proposed metrics. NODE with decomposition, which coincides with other proposed metrics.

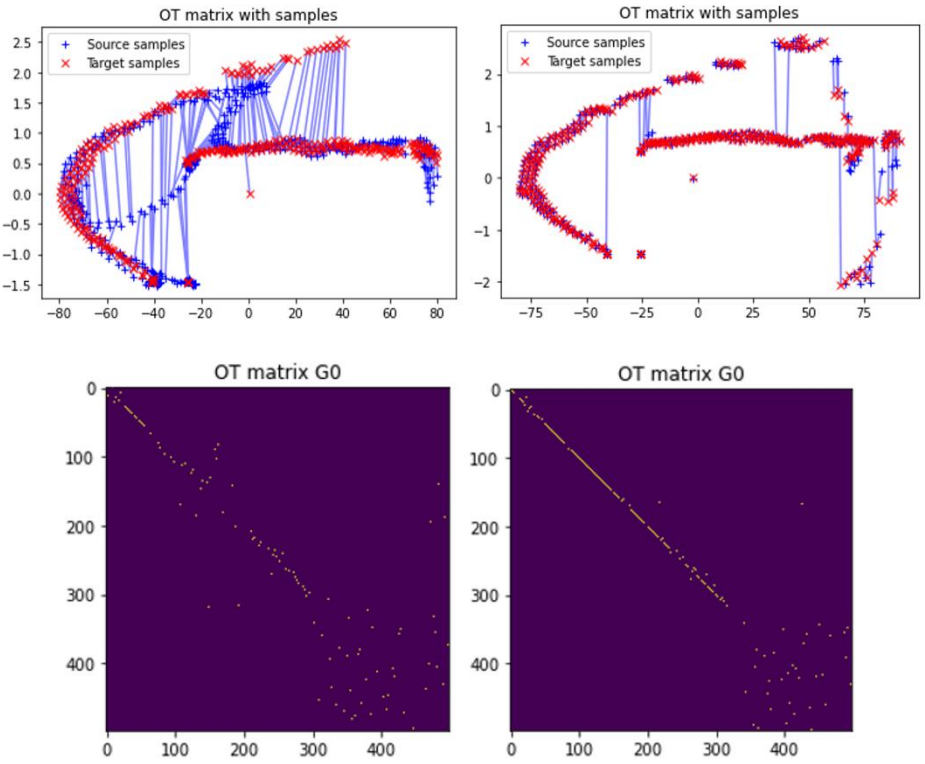

Fig. 5. OT for a given timestep, DMD (on the left) and NODEwD (on the right)

In the second row yellow dots correspond to pairs of neighbors (that is, if particles i and j were considered neighbors after the minimization step, the intersection of i^{th} row and j^{th} column is yellow). As one can notice, in case of NODEwD most of the dots are located near the diagonal, which leads to conclusion that this model does not mix particles much, thus the swapping situations do not occur. DMD, in contrast, has many mixing occurrences, which supports the assumption that local dynamics is not captured well.

5.2 Models Performance

In the test dataset the configuration of emitters is the same as in train dataset. For validation dataset the position and direction of emitters were changed. Liquid properties remained the same for train, test and validation datasets.

Uniform MSE for DMD has its tail not exceeding 60. The row-wise MSE does not exceed 1.5, however, it has two distinct peaks, the second one close to the maximal value. The supremum metric is bounded by 80, and each timestep has outliers with deviations from true data comparable to the upper bound for coordinates. DMD library [18] used in the project does not provide the researcher with means on testing the model on validation data. When DMD is used, the shift operator A, described in

section 2.1, must be recomputed each time new data is added, it is senseless to speak about validation on new dataset – it would impact the model and make any comparison incorrect.

For GNN uniform MSE has a tail longer than for DMD, indicating the presence of particles with coordinates deviating from true ones up to 160. Row-wise MSE has a distinct peak biased to the right, hence errors for timesteps are distributed less uniformly than for DMD. Supremum metrics indicates the presence of deviations of size up to 160. Although the performance of GNN on validation dataset is worse than that on the test one, metrics do not increase as much as for other models. The length of the tail for Uniform MSE increases by a factor of two, the same for supremum metric. And row-wise MSE loses its double-peaked structure, remaining bounded by 1.7

Uniform MSE for NODE has a tail much shorter than for GNN and DMD, upper-bounded by 7. Row–wise MSE does not exceed 0.06, which is about 100 times smaller than for the two models described earlier. The supremum metric has two distinct peaks and is bounded by 7. The Row-wise MSE for validation dataset is significantly worse: it increases by a factor of 10. The supremum metric, along with the Uniform MSE, is upper bounded by 300. Notice that this increase in metrics exceeds that for GNN.

Although NODEwD has a tail for uniform MSE stretching up to 160, it is lighter than for the first three models. Row-wise MSE is upper-bounded by 0.35, thus exceeding the one for NODE. The supremum metric indicates performance similar to that of GNN. However, after validating the model on new dataset one obtains more peaked structure of histograms, which indicates the presence of significant part of particles deviating from their true positions.

CNNwD has characteristics similar to those of NODEwD. Thus, Uniform metric has the same upper bound and tail mass. Row-wise MSE, again, does not exceed 0.35, and the supremum metric is bounded by 160. The validation results indicate that the increase in Uniform MSE and supremum metrics is larger than that for the previous model. However, the Row-wise MSE is only multiplied by a factor of 2, which is two times smaller than for NODEwD.

The results for the regular models' MSE are summarized in Table 3. For each model the average performance for different boundary and initial conditions is measured, along with margin of error. However, it is easy to notice that the MSE is comparably small to the absolute values of the particle coordinates, hence the data-driven predictions are robust to perturbations of the system.

Table 3. MSE for different models with margin of error based on all configurations

	DMD with *rk*		NODEwD, *rk*		CNNwD, *rk*		NODE	GNN
	5	20	5	20	5	20		
MSE	1.45 ±0.35	0.72 ±0.25	0.64 ±0.14	0.19 ±0.10	0.58 ±0.20	0.23 ±0.15	0.21 ±0.10	0.35 ±0.05

One can see that results presented in the Table 3 correspond to those described in the paper earlier. DMD has the worst indicators of performance for all decomposition ranks. The lowest MSE is obtained by NODEwD, which is followed by NODE and

CNNwD. However, the decomposition models require higher ranks for this level of accuracy. The MSE for GNN is comparable to this of CNNwD and NODEwD, being much lower than that of DMD.

The best picture of particle trajectories is provided by NODEwD and CNNwD. All the directions are preserved, and only small fluctuations can be noticed. This is not unexpected, as these two approaches operate with global data, omitting small details of dynamics which can affect picture if poorly reconstructed. The GNN and NODE generate pictures which may look similar to each other. Here fluctuations of individual particles do not allow one to discern the global movement patterns, thus hiding main directions of dynamics. The trajectories generated by DMD is smoother than the true one. Even though some main patterns in movement are preserved, smaller fluctuations are not taken into account. However, for some problems this may be useful, since main directions of movement may be most important for qualitative analysis of the system.

The last thing to mention when comparing results obtained by our models is training and inference time. The fastest model is DMD, both in training and in inference, both times do not exceed quarter of second. This is not unexpected, as only matrix operations are needed for this model. Of all the other model the fastest one is GNN. It requires about 10 minutes for training and 2 minutes for inference. However, in order to use it one needs to construct a neighbor graph, which is a computationally extensive procedure. NODE and CNN are the slowest models, requiring 10 minutes for training and 5 for inference. The inclusion of decomposition reduces this time to like that of GNN, which results from data preprocessing applied on the decomposition step. With the graphical processing unit with 32GB of video RAM used for neural networks training and inference, the decrease in training and inference time for all the applicable models by a factor of 5 was achieved.

6 Conclusions

Modeling complex systems even as simple as 2D liquid simulator is a daunting task, which requires significant expertise, hand-crafted work and major time investment. The advent of data-driven approaches enables to overcome some of these issues. In this paper results of comparison of data-driven models for 2D liquid simulation were provided. There exist three possible problem statements, which are one-step prediction, rollout prediction and inverse modeling. Data-driven methods were split into global and local approached. The first one measures overall model performance, neglecting large deviations in behavior of individual particles as soon as they don't affect the whole system. The local approach was concerned with movement of individual particles. It may be viewed as the measure of worst-case performance, or just the only possible way of comparing models when the highest precision possible is needed.

Several types of metrics were introduced to evaluate the models' performance. First category of metrics deals with MSE, and evaluates the performance uniformly, row-wise, supremum and for the whole system at once. The second category of met-

rics deal with distributions and allows to work with situations when particles interchange their position, so that the whole picture does not change, but MSE is large. Also, trajectories for models were depicted together. Metrics of performance for data-driven model are better to be used together, since categories outline different aspects of performance. The results for a model can be better for first metric and often turn out to be much more modest for another metric compared to another model performance.

For the one-step prediction problem NODE with decomposition provide best overall results. The application of decomposition significantly reduces the complexity of the system studied, although some effects (such as emergence of new patterns) are sometimes missed. Combining continuous models with decomposition provides best overall results for one-step prediction problem.

Extrapolating model trained on a system with one set of parameters for liquid and emitters to a different case is not accurate and additional procedures need to be applied to improve performance. This happens due to the fact methods which give higher precision use decomposition, which is very sensitive to changes in parameters. Methods which provide better generalization abilities are not the most precise ones.

Although, the problem of rollout prediction is out of scope of this work, required model accuracy needs to be higher than for one-step prediction, as even small deviations in predictions result in system destabilization after few timesteps. Main directions of further research include the survey and application of new metrics, which fully describe different aspects of liquid behavior. Another important step for model generalization is the study of noise impact, which would be important for production use of simulators and working with real noisy data. Since models have their own weaknesses, the construct of an aggregated model can lead to better performance combining benefits of different approaches and eliminate their weaknesses.

References

1. Schmid, P. J. Dynamic mode decomposition of numerical and experimental data. Journal of fluid mechanics 656, 5-28 (2010).
2. Sanchez-Gonzalez, A., Heess, N., Springenberg, J.T., Merel, J., Riedmiller, M., Hadsell, R., Battaglia, P. Graph networks as learnable physics engines for inference and control. In: Proceedings of the 35th International Conference on Machine Learning, PMLR vol. 80, pp. 4470-4479, Stockholm (2018).
3. 2D Liquid Simulation in WebGL, https://github.com/Erkaman/gl-water2d, last accessed 2021/8/31.
4. Rojas, C. J., Dengel, A., Ribeiro, M. D. Reduced-order Model for Fluid Flows via Neural Ordinary Differential Equations. arXiv preprint arXiv:2102.02248, (2021).
5. Portwood, G.D., Mitra, P.P., Ribeiro, M.D., Nguyen, T.M., Nadiga, B.T., Saenz, J.A., Chertkov, M., Garg, A., Anandkumar, A., Dengel, A. Baraniuk, R. Turbulence forecasting via Neural ODE. arXiv preprint arXiv:1911.05180, (2019).
6. Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., Battaglia, P.: Learning to simulate complex physics with graph networks. In: Proceedings of the 37th International Conference on Machine Learning, PMLR vol. 119, pp. 8459-8468, Vienna (2020).

7. Yan, J., Mu, L., Wang, L., Ranjan, R., Zomaya, A. Y. Temporal convolutional networks for the advance prediction of ENSO. Scientific reports 10, 8055 (2020).
8. Shlomi, J., Battaglia, P., Vlimant, J. R. Graph neural networks in particle physics. Machine Learning: Science and Technology, 2(2), 021001, (2020).
9. Murata, T., Fukami, K., Fukagata, K. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. Journal of Fluid Mechanics 882, A13, (2020).
10. Bonnaffé, W., Sheldon, B. C., Coulson, T. Neural ordinary differential equations for ecological and evolutionary time- series analysis. Methods in Ecology and Evolution 12(7), 1301-1315 (2021).
11. Brunton, S. L., Budišić, M., Kaiser, E., Kutz, J. N. Modern Koopman theory for dynamical systems. arXiv preprint arXiv:2102.12086, (2021).
12. Birdi, K. A. S. Handbook of surface and colloid chemistry. 3rd edn. CRC press, Boca Raton (2008).
13. Chen, R. T., Rubanova, Y., Bettencourt, J., Duvenaud, D. Neural ordinary differential equations. In: Proceedings of the 32nd International Conference on Neural Information Processing System, pp. 6572–6583, Montreal (2018).
14. Li, Z., Yang, W., Peng, S., Liu, F. A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Transactions on Neural Networks and Learning Systems, 1-21, (2020).
15. Villani, C. Optimal transport: old and new. Springer Science & Business Media, Berlin (2008).
16. Müller, A. Integral probability metrics and their generating classes of functions. Advances in Applied Probability 29(2), 429-443 (1997).
17. Billingsley, P. Convergence of probability measures. 2nd edn. John Wiley & Sons, New York (2013).
18. Demo, N., Tezzele, M., Rozza, G. PyDMD: Python dynamic mode decomposition. Journal of Open Source Software 3(22), 530 (2018).