Convolutional Recurrent Autoencoder for Molecular-Continuum Coupling

Piet Jarmatz[0000−0002−5463−0740], Sebastian Lerdo[0000−0001−5148−7056], and Philipp Neumann[0000−0001−8604−8846]

Chair for High Performance Computing, Helmut Schmidt University Hamburg, Germany, jarmatz@hsu-hh.de

Abstract. Molecular-continuum coupled flow simulations are used in many applications to build a bridge across spatial or temporal scales. Hence, they allow to investigate effects beyond flow scenarios modeled by any single-scale method alone, such as a discrete particle system or a partial differential equation solver. On the particle side of the coupling, often molecular dynamics (MD) is used to obtain trajectories based on pairwise molecule interaction potentials. However, since MD is computationally expensive and macroscopic flow quantities sampled from MD systems often highly fluctuate due to thermal noise, the applicability of molecular-continuum methods is limited. If machine learning (ML) methods can learn and predict MD based flow data, then this can be used as a noise filter or even to replace MD computations, both of which can generate tremendous speed-up of molecular-continuum simulations, enabling emerging applications on the horizon.
In this paper, we develop an advanced hybrid ML model for MD data in the context of coupled molecular-continuum flow simulations: A convolutional autoencoder deals with the spatial extent of the flow data, while a recurrent neural network is used to capture its temporal correlation. We use the open source coupling tool MaMiCo to generate MD datasets for ML training and implement the hybrid model as a PyTorch-based filtering module for MaMiCo. It is trained with real MD data from different flow scenarios including a Couette flow validation setup and a three-dimensional vortex street. Our results show that the hybrid model is able to learn and predict smooth flow quantities, even for very noisy MD input data. We furthermore demonstrate that also the more complex vortex street flow data can accurately be reproduced by the ML module.

Keywords: Flow Simulation · Machine Learning · Denoising · Data Analytics · Molecular Dynamics · Molecular-Continuum

1 Introduction

One of the fundamental nanofluidics tools in engineering, biochemistry and other fields are molecular dynamics (MD) simulations [5, 14]. MD has the potential to assess the properties of novel nanomaterials, for example it has been applied to water desalination, in order to develop highly permeable carbon nanotube

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

membranes for reverse osmosis [20]. However, in many such application cases the challenging computational cost of MD imposes a barrier that renders full-domain simulations of larger nanostructures infeasible. Thus, coupled multiscale methods, where MD is restricted to critical locations of interest and combined with a continuum flow model to create a molecular-continuum simulation, are typically used to enable computationally efficient simulations of macroscopic flows. In the water purification example, the MD simulation of the carbon nanotube confined flow can be accompanied by a Hagen-Poiseuille flow solver to yield a multiscale method that still captures molecular physics but also makes water transport predictions for larger laboratory-scale membranes [2].

Here we focus on cases where the flow in the MD domain could be described by the continuum solver, unlike carbon nanotubes, so that we can use the additional continuum information to validate coupling methodology. Since from a software design perspective, implementing a coupling can come with many challenges, many general-purpose frameworks for arbitrary multiphysics simulations are available (e.g. [3, 19, 21]), however only a few focus on molecular-continuum flow [8, 15, 18]. In our recent work [8] we have presented MaMiCo 2.0, an open source C++ framework designed to create modular molecular-continuum simulations. Data sampled from MD often suffers from high hydrodynamic fluctuations, i.e. thermal noise, but many coupling schemes and continuum methods depend on smooth flow data. Hence, MaMiCo supports two ways to obtain smoother data: ensemble averaging and noise filtering. For the former, an ensemble of independent MD simulation instances is launched on the same subdomain, so that their results can be averaged [13]. Depending on the temperature, typically 50-200 instances are necessary to obtain a stable transient two-way coupled simulation [23]. For noise filtering, there is a flexible filtering system with several filter module implementations, such as proper orthogonal decomposition (POD) or Non-Local Means (NLM) [6]. NLM can reduce the number of MD instances required for a certain flow result accuracy, approximately by a factor of 10, for details see [6]. MaMiCo 2.0 can change the number of active MD instances dynamically at run time of the coupled simulation for on-the-fly error control [8].

However, also with a combination of ensemble averaging to obtain averaged data for stable coupling and noise filtering to reduce the number of MD instances, a problem of major importance remains to be the computational cost of MD, which restricts the applicability of the methodology to limited scenarios. A promising approach to further tackle this are machine learning (ML) methods which can support the coupling in this case in two ways: First, if ML is used as an advanced noise filter in the filtering system of MaMiCo, then this can help to reduce the number of MD instances. Second, if an ML model is able to learn and predict the behavior of MD, then it can be used to avoid costly computations and to replace the MD simulation, at least for some of the time steps or some of the instances. Note that from ML perspective, both use cases are very similar: in either case, the ML module receives as input data the state of MD, i.e. the data sampled from MD at coupling time step t, as well as the information from the

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

continuum solver at the outer MD boundaries, and the ML module is supposed to output only the filtered MD state, either at t or at $t+1$. Note that while these use cases are the primary motivation for the ML developments presented in this paper, here we will focus on definition, training, validation and analysis of the novel ML module for the purposes of both filtering and prediction of the MD flow description, yet abstaining from replacing MD entirely which we consider a natural next step in the near future, since this is an active field of research. For instance, in [9] a surrogate model of MD simulations is developed, which can accurately predict a small number of key features from MD, based on a fully connected neural network. Here we focus on architectures for grid-based flow data instead, since fully connected networks are not scalable to operate on large inputs.

In recent years, many works have studied various neural network architectures for complex fluid flow and turbulent CFD problems [4, 12]. For instance, Wiewel et al. have developed a hybrid model using a convolutional neural network (CNN) based autoencoder (AE) and an LSTM, which is a type of recurrent neural network (RNN) [22]. While they achieve significant speed-ups compared to traditional solvers, they did not investigate the impacts of noisy input data. However, in 2022, Nakamura and Fukagata have performed a robustness analysis, investigating the effects of noise perturbation in the training data in the context of an CNN-AE for turbulent CFD [11], although without employing an RNN. Here, we combine and use insights from both these works by applying a similar hybrid model to a different context, molecular dynamics data, which often contains a higher level of noise.

The goal of this paper is to introduce a convolutional recurrent hybrid model and to demonstrate that it is able to do both: work as an advanced MD noise filter and accurately predict the behavior of the MD simulation over time in a transient three-dimensional molecular-continuum flow.

In Section 2 we introduce the coupling methodology, software tools, flow solvers and scenarios used in this paper. In Section 2.1 we give details how they are applied to generate the datasets for the ML training. Section 3 develops a convolutional recurrent autoencoder model for MD data. Therefore, first we define an AE in Section 3.1, then we introduce an RNN for the CNN latent space in Section 3.2, and finally in Section 3.3 we combine both of them into a hybrid model. Section 4 gives details about our implementation of this hybrid model, documents the training approach and defines hyper-parameters. We first show results of the hybrid model for a Couette flow validation test case in Section 5, and then demonstrate and explain its capabilities for a more complex vortex street in Section 6. Finally, Section 7 summarizes our insights and provides an outlook to future research that is rendered possible by this work.

2 Molecular-Continuum Coupled Flow

For our molecular-continuum simulations we consider a domain decomposition into a small molecular region placed within a much larger continuum domain,

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

with nested time stepping between the two solvers. For the datasets used in this paper, on the particle side MaMiCo's in-house MD code *SimpleMD* is applied to model a Lennard-Jones fluid and on the continuum side we use the *lbmpy* [1] software package to apply a Lattice-Boltzmann method (LBM).

In this paper, as illustrated in Fig. 1, we use a Couette flow case similar to *scenario 2A* defined in [6] and a vortex street test case similar to *scenario 1*, corresponding to the benchmark *3D-2Q* by Schäfer et al. [17]. The coupling methodology, software and flow test scenario setup have been described thoroughly in previous publications, thus the reader is referred to [6, 7] for more details. The Kármán vortex street (KVS), while not a common nanofluidics scenario, is excellent for investigating and comparing the performance of ML models, since it yields a sufficiently complex multidimensional transient flow with challenging non-linear data. In both scenarios we zero-initialize the flow. We start to couple with MD during a transient start-up phase. Since coupling is already enabled while the continuum solver is establishing the flow, the velocities vary more. This can be used to reveal potential mismatches between MD data and ML predictions. The tests performed in this paper use a single MD instance only, for simplicity and because this is the most challenging test case for an ML module, with the highest possible level of noise in the data sampled from MD. All simulations used in this paper are *one-way* coupled, meaning that flow data is transferred from the continuum solver to the particle system, but not in the other direction.

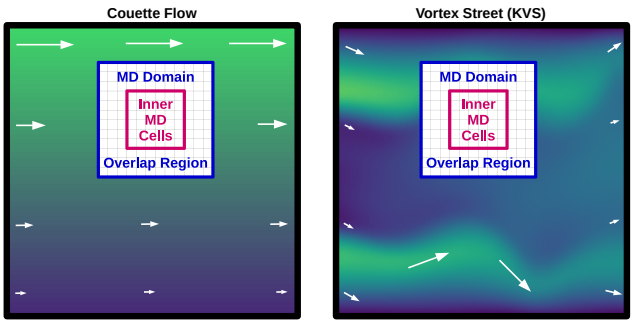


Fig. 1. Transient coupled flow scenarios; MD receives continuum data in overlap region, ML aims to predict MD data in inner cells. Left: flow between two parallel plates. Right: vortex street, obstacle not shown.

2.1 Dataset Creation

The filtering system of MaMiCo operates on a regular grid of voxel cells that covers the MD domain, where each cell contains quantities, such as density, temperature or velocity, which are sampled from the MD simulations and averaged over all particles within that cell. Typically each cell contains ca. 10 to 100 particles, depending on the exact simulation configuration. Note that since we train and use ML models as modules in the filtering system of MaMiCo, they do not have any access to individual particles, but only to grid-based quantities. This means that the ML model does not operate on the full state of the MD system, but instead on the information that is exchanged between MD and continuum solver. If ML can predict this information with a sufficient accuracy, then it can

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

replace the MD simulation from the perspective of the continuum solver – even without any knowledge of the internal MD state.

In this paper, the ML training and validation datasets generated by MaMiCo are multichannel volumes in the shape of $[900 \times 3 \times 24 \times 24 \times 24]$. The first dimension refers to 900 coupling cycles, i.e. macroscopic simulation time steps, while the second dimension refers to three flow velocity components per voxel, u_x, u_y and u_z. For simplicity, we disregard the other quantities stored in the cells, but the ML methodology can be applied to them in an analogous way. The remaining dimensions define the volume of interest spanning $24 \times 24 \times 24$ cells covering the entire MD domain. There is an overlap region that covers the three outer cell layers of the MD domain, where data from the continuum solver is received and applied to the MD system as a boundary condition, using momentum imposition and particle insertion algorithms, for details see [6]. On the inner MD domain consisting of $18 \times 18 \times 18$ cells, quantities are sampled from MD for transfer to the continuum solver. The goal of the ML model developed in Section 3 is to predict a future filtered state of these quantities in the inner MD domain, excluding the overlap region, while given as input a noisy state from past coupling time steps of the entire domain, including the overlap region. This means that the ML model can access the data that is or would be transferred from the continuum solver to MD, and in turn tries to replace both, MD simulation and noise filters, in order to forecast the data that would finally be transferred back to the continuum solver one coupling cycle later.

Our data stems from 20 KVS and 21 Couette flow MaMiCo simulations. To generate this amount of data, we vary three parameters and choose seven Couette wall velocities at three MD positions, and five KVS init-times at four MD positions. For example, a KVS dataset where the coupling starts after 22000 LBM steps and the MD domain is placed north-west of the KVS center is labeled as '22000_NW' (compare Fig. 6). The most important parameters defining the simulations in this paper are: 175616 molecules per instance, density $\rho \approx 0.813$, Lennard-Jones parameters $\sigma_{LJ} = \epsilon_{LJ} = 1$, cutoff radius $r_{cutoff} = 2.5$, temperature $T = 1.1$, yielding a kinematic viscosity $\nu \approx 2.63$, given in dimensionless MD units (see [6]). In the KVS scenario, we couple with 50 MD steps per cycle to a D3Q19-TRT LBM on 12580620 LB cells, and place the center of MD at $50\% \times 73.2\% \times 50\%$ of domain size in *3D-2Q* setup, plus small variations. All of this yields a data size of 12.2 *GB* (in binary format), we split it into 80% training and 20% validation data sets. More detailed information generating the datasets can be accessed online[1].

3 Convolutional Recurrent Autoencoder

In this section, we introduce our ML model developed to make time-series predictions of microscale flow velocity distributions. Designing such a model must reflect the spatial and temporal dependencies in the underlying physics [4, 22].

[1] https://github.com/HSU-HPC/MaMiCo_hybrid_ml

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

Advances in computer vision have shown that spatially correlated data, such as images or volumes, are best dealt with CNNs [10]. Advances in machine translation and speech recognition have shown that sequentially correlated data such as language can be modeled with RNNs [24]. Our models follow a hybrid approach combining their advantages. Our experiments have shown that in this MD data context, a classical AE performs better than a model based on the *U-Net* architecture [16]. Thus, in the following we focus on a concept similar to the approach presented by Nakamura et al. [12].

3.1 Convolutional Autoencoder (AE)

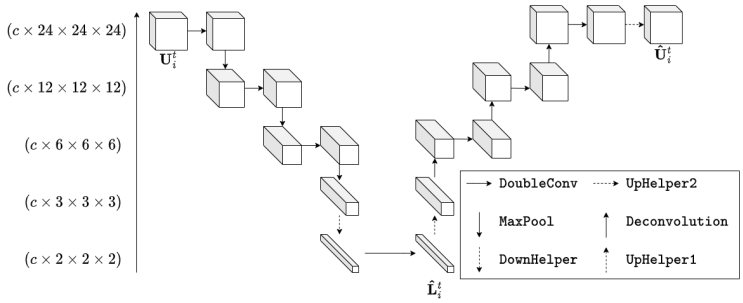


Fig. 2. Schematic of a single CNN AE as used for the triple model approach.

A convolutional AE is a type of CNN that consists of an encoding and a decoding path, as shown in Fig 2. It aims to learn a dense comprehensive representation of the input. The sizes of our datasets make RNN approaches impractical, thus we use an AE to encode lower-dimensional representations of the input, called latent space [12]. Let $\mathbf{U}$ be the velocity distribution input data and $h(\mathbf{U})$ describe the encoding function. The encoding function $h(\mathbf{U})$ maps the input data $\mathbf{U}$ to a latent space representation $\hat{\mathbf{L}}$, using convolutional and pooling layers such that

$$h(\mathbf{U}) = \hat{\mathbf{L}} = \sigma(\mathbf{W}_h \mathbf{U} + \mathbf{b}_h) \, . \tag{1}$$

Here, $\mathbf{W}_h$ and $\mathbf{b}_h$ are the weights and biases of the encoding function, and σ is a non-linear activation function, such as ReLU. The decoding function $g(\hat{\mathbf{L}})$ maps $\hat{\mathbf{L}}$ back to the original input space $\hat{\mathbf{U}}$ using transposed convolutional and upsampling layers such that

$$g(\hat{\mathbf{L}}) = \hat{\mathbf{U}} = \sigma(\mathbf{W}_g \hat{\mathbf{L}} + \mathbf{b}_g) \, . \tag{2}$$

Here $\mathbf{W}_g$ and $\mathbf{b}_g$ are the weights and biases of the decoding function. The AE is trained by minimizing a reconstruction loss function $\mathcal{L}(\mathbf{U}, \hat{\mathbf{U}})$ which measures the difference between the original input $\mathbf{U}$ and the reconstructed input $\hat{\mathbf{U}}$. The models used in this paper follow a single or triple model approach. The former uses a single AE to operate on the entire velocity distribution $\mathbf{U}$ while the triple model uses three identical instances of an AE designed to operate on a single

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

velocity channel $\mathbf{U}_i$. For each AE, in the encoding path, the first three horizontal groups apply a `DoubleConv` layer consisting of two ReLU-activated 3×3 same convolutions followed by a `MaxPool` layer consisting of a 2×2 max pooling operator. This operator halves each spatial dimension. Next, a `DownHelper` consisting of a ReLU-activated 2×2 valid convolution is applied in order to further reduce dimensionality. Finally, another `DoubleConv` layer is applied thereby yielding the latent space $\hat{\mathbf{L}}_i$. In the decoding path, $\hat{\mathbf{L}}_i$ is first passed to an `UpHelper1` consisting of a 2×2 deconvolution. Next, a `Deconvolution` tailored to double the spatial dimensions followed by a `DoubleConv` is applied. This is repeated until the input spatial dimensionality is restored. Finally, an `UpHelper2` consisting of a 3×3 same convolution tailored to a single channel output is applied. When the output of the AE is used in the hybrid model, only the inner 3×18^3 values are selected and passed on towards the continuum solver, since they correspond to the inner MD cells excluding the overlap layers.

3.2 Recurrent Neural Network

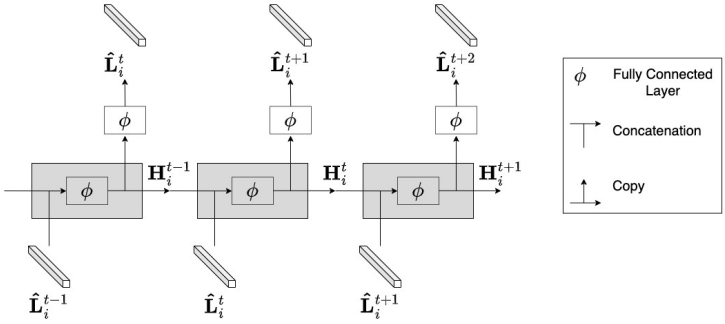


Fig. 3. Schematic of one unrolled RNN as used for the time-series prediction of microscale fluid flow latent spaces.

We account for the temporal dependency via an RNN with a hidden state as depicted in Fig 3. The hidden state at a given time step is a function of the hidden state of the previous step and the input of the current step. With this, the hidden state is able to preserve historical information from previous time steps of the sequence. Let $\mathbf{H}^t$ be the hidden state at time t, given as [25]

$$\mathbf{H}^t = \sigma(\hat{\mathbf{L}}^t \mathbf{W}_{xh} + \mathbf{H}^{t-1} \mathbf{W}_{hh} + \mathbf{b}_{hh}) \, . \tag{3}$$

Here, $\mathbf{W}_{xh}$ are the input weights, $\mathbf{W}_{hh}$ are the hidden state weights and $\mathbf{b}_{hh}$ are the biases. The RNN latent space output $\hat{\mathbf{L}}^{t+1}$ is then the output of a fully connected layer such that

$$\hat{\mathbf{L}}^{t+1} = \mathbf{H}^t \mathbf{W}_{hq} + \mathbf{b}_{qq} \, . \tag{4}$$

$\mathbf{W}_{hq}$ and $\mathbf{b}_{qq}$ are output weights and biases. A single or triple AE requires a similar RNN approach. This means one RNN operates on $\hat{\mathbf{L}}$ in a single model,

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

while the triple model requires three RNNs to each operate on one component $\hat{\mathbf{L}}_i$. Fig. 3 depicts such an RNN that is a component of a triple model.

3.3 Hybrid Model

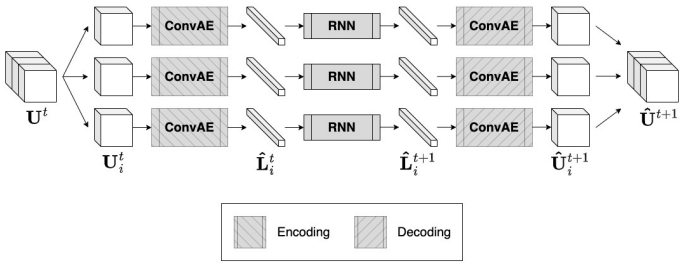


Fig. 4. Schematic of the convolutional recurrent autoencoder as employed in the triple model approach for the task of time-series prediction of cell-wise averaged fluid flow velocities.

Our two hybrid model architectures combine the single and triple model convolutional AEs and RNNs. Figure 4 depicts the hybrid triple model where the input $\mathbf{U}^t$ corresponds to the multichannel volume at t. The input channels $\mathbf{U}_i^t$ are separated and propagate independently of each other through the network. First, the AE determines the corresponding latent space representations $\hat{\mathbf{L}}_i^t$. This is then passed to the RNN to predict the corresponding latent space representations of the next time step $\hat{\mathbf{L}}_i^{t+1}$. Next, the predicted latent spaces are passed back to the AE thereby predicting the single channel velocity distributions $\hat{\mathbf{U}}_i^{t+1}$ for $t+1$. Combining all the $\hat{\mathbf{U}}_i^{t+1}$ yields the multichannel velocity distributions $\hat{\mathbf{U}}^{t+1}$ at $t+1$. Note that since the flow physics is invariant to rotations, the three models are the same. The hybrid single model works in the same way, except that it does not separate the input channels.

4 Implementation and Training Approach

We implement the convolutional recurrent hybrid model for MaMiCo using the open-source machine learning framework PyTorch[2]. PyTorch offers CUDA support for GPU-based deep learning. In the following we briefly describe our training approaches for the AE and the RNN. As [12, 22] show, they can be trained separately. Both the single and triple model approaches require to first train the AE by itself. After having successfully trained the AEs, the latent spaces can be generated from the original datasets. Then the RNN can be trained on them. The RNN must be trained on the same AE it is actually used with, i.e. if the AE ever changes, then the RNN has to be trained again. For brevity, the training configurations and hyper-parameters are presented in Table 1. As listed there,

[2] https://pytorch.org/docs/stable/index.html

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

we apply the single model to the Couette scenario and the triple model to the KVS scenario, because only the KVS dataset shows a multidimensional transient flow.

Table 1. Training Configurations

	Conv. AE		**RNN**	
Model	Single	Triple	Single	Triple
Dataset	Couette	KVS	Couette	KVS
Loss, Activ. Fn.	MAE, ReLU	MAE, ReLU	MAE, tanh	MAE, tanh
Optimizer	opt.Adam	opt.Adam	opt.Adam	opt.Adam
Batch Size	32	32	32	32
Epochs, Learn. Rate	250, 1e-4	100, 1e-4	250, 1e-4	15, 1e-4
#Layers, Seq. Size	-	-	1, 25	1, 25
Shuffled, Augmented	True, False	True, True	True, False	True, True

Convolutional Autoencoder (AE) The triple models are trained with augmented versions of the velocity resulting from swapping the channels of the original datasets, i.e. the permutations $(\mathbf{U}_0, \mathbf{U}_1, \mathbf{U}_2)$ and $(\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_0)$ and $(\mathbf{U}_2, \mathbf{U}_0, \mathbf{U}_1)$ are used. This is done so that the models are encouraged to learn a more general mapping by means of a greater variance in the inputs.

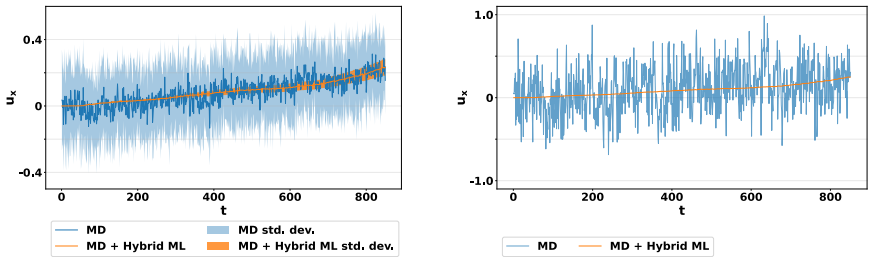
RNN The RNN takes a sequence of the past 25 latent spaces $[\hat{\mathbf{L}}^{t-24}, \ldots, \hat{\mathbf{L}}^{t}]$ and performs the time-series prediction to yield an estimate for $\hat{\mathbf{L}}^{t+1}$. We choose a sequence length of 25 here, because in our experiments that performed best at minimizing validation loss, e.g. it was 34% better compared to a sequence length of 15 latent spaces. A range of about 10-50 is generally reasonable because it should be long compared to the frequency of fluctuations in MD data, but short compared to simulation run time.

In contrast to more common approaches where model prediction is sanctioned by means of loss quantification w.r.t. to the model target output, i.e. $\mathcal{L}(\hat{\mathbf{L}}^{t+1}_{i,\text{targ}}, \hat{\mathbf{L}}^{t+1}_{i,\text{pred}})$, we implement a loss quantification in the velocity space by comparing the decoded latent space prediction to the target single channel velocity distribution, i.e. $\mathcal{L}\big(\mathbf{U}^{t+1}_i, g(\hat{\mathbf{L}}^{t+1}_{i,\text{pred}})\big)$. This helps to train the RNN in such a way that the combined hybrid model, including the CNN decoder, minimizes the error in its predicted flow velocities.

5 Results – Couette Flow Scenario

In order to validate the relatively simple single model hybrid ML approach, we choose a Couette flow start-up test scenario as defined in [6]. There is no macroscopic flow in Y and Z directions, thus we focus on the direction of the moving wall, i.e. the X component of the velocity u_x, which is shown in Figure 5 over 850 simulation time steps, i.e. excluding a 50 step initialization phase. Figure 5a averages u_x over a line of cells and displays its standard deviation over these cells as a lighter shaded area, while Figure 5b shows the value for one cell only,

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42



(a) Flow velocity u_x over time in a line of 24 cells through MD domain

(b) Flow velocity u_x over time in one cell centered in MD domain

Fig. 5. Comparison of CNN+RNN hybrid ML model predictions in orange with raw MD data in blue for x-component of flow velocity u_x in Couette flow scenario.

which is in the center of the MD domain. It can be observed that the raw MD data exhibits a high level of random noise, caused by thermal fluctuations. Here we investigate the noise filtering properties of the ML model. In Fig. 5a one can notice that the standard deviation of the ML output grows slightly over time (i.e. the orange shaded area gets wider). However, it can be seen in both Fig. 5a and 5b that the hybrid ML predictions constitute a very stable noiseless signal, that are in good agreement with the mean flow displayed by the fluctuating MD data. Note that the ML model was not trained with information from a continuum flow solver, and also not with any noise filtering algorithm, instead it was trained on noisy raw MD data only (test case 'C_1_5_M' in the online repository). The filtering effect seen in Figure 5 is obtained in space due to the dimensionality reduction into the latent space of the CNN AE and in time due to the application of the RNN (shown separately below, see Fig. 6 and 7), thus the desired filtering effect is already designed into the architecture of the hybrid model.

6 Results – Kármán Vortex Street Scenario

To predict more complex flow patterns, we set up a vortex street scenario (KVS). It exhibits multidimensional non-steady signals in each of the flow velocity components, so that the ML performance can be investigated adequately. Figure 6 compares the ML predictions with raw MD data, for one of the cells in the center of the MD domain. Both Figures 6 and 7 evaluate the ML models on test cases from the validation set, i.e. on data on which they have **not** been trained. Fig. 6 shows the performance of the AE only, i.e. the result of en- and decoding MD data. This aims to help the reader distinguish the CNN and the RNN impacts on the hybrid model behavior. It can be seen in Fig. 6 that the AE is able to represent all information necessary to capture and preserve the mean flow characteristics, while it does not preserve spatial noise present in the MD data (i.e. visually that the orange curve follows only a sliding mean of the blue curve). However, it can also be seen that the raw AE output is more noisy

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

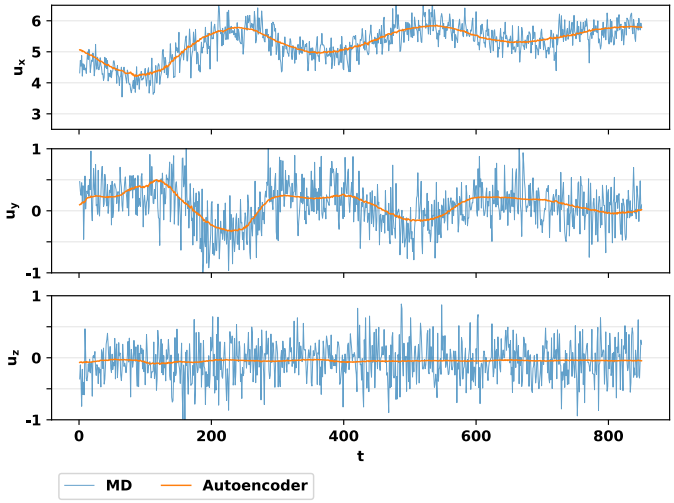


Fig. 6. AE vs. MD, KVS validation '22000_NW', for one cell in MD domain center

than the hybrid model data (i.e. orange curve less smooth than in Fig. 7), as the temporal smoothing and prediction coming from the RNN is missing here.

In contrast to that, Fig. 7 shows the CNN+RNN hybrid ML model. For the orange curve, MD data is used as ML input, similar to Fig. 5, a filtering effect is obtained. But for the green curve, the hybrid ML model is used standalone, without any MD input data. The ML input is zero-initialized. The effect of this can be seen for small values of t in Fig. 7, where the green curve starts close to zero. Then the ML model is applied recursively, so that all of its inputs for inner MD cells are its own previous outputs. For the overlap cell region, the ML input is coming from the continuum solver, meaning that the ML model receives the same information as a coupled MD. Note that we plot a cell in the domain center, far away from this overlap region, so that it is not directly influenced by continuum data. Instead the plot shows the time evolution of the ML data during a long series of recursive evaluations. This is very different from the MD+ML case (orange curve) where the ML model gets real MD input data in every time step and performs a prediction for only one step into the future. Since no MD data is fed into the ML model here, there is less confidence that the model will stick to physical MD behavior, especially over longer time spans. However, except of just two small offsets for u_y around $t = 250$ and $t = 550$, it is still in an excellent agreement with the correct particle data, revealing the prediction capabilities of the hybrid model.

To give more details about the computational performance of these surrogate model predictions, compared to the original full-scale MD: the AE training in this case took ten hours on a single NVIDIA A100 GPU, the RNN training one hour, yielding eleven hours in total for the hybrid model training. The evaluation of the trained model in our experiments on an Intel Xeon 8360Y CPU costs 81.7 ms, while running the MD model sequentially on the same CPU requires about 19 200 ms per coupling cycle.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

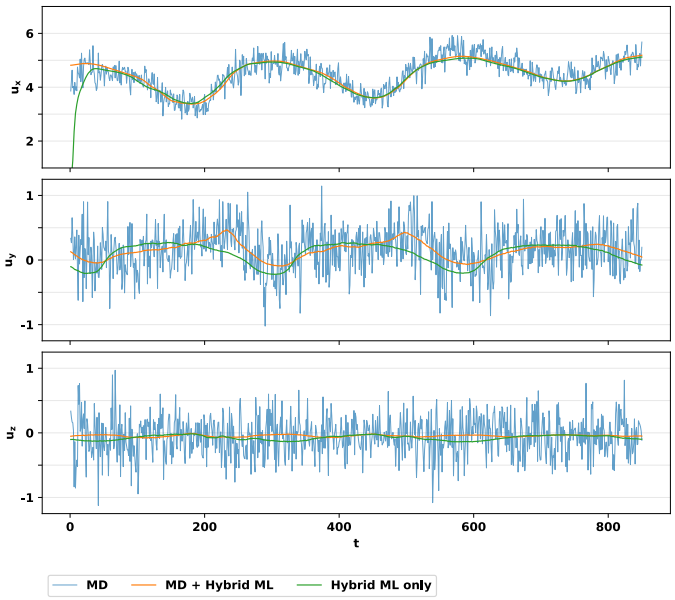


Fig. 7. Raw MD vs. CNN+RNN hybrid ML model performance with and without MD input data, KVS validation case '20000_NE', for one cell in center of MD domain

Figure 8 presents the same insights as Fig. 7, but in a greater level of detail. It shows the same test case, i.e. flow data on which the ML models have **not** been trained. Here, instead of only one cell, a line of 24 cells in the MD domain is plotted, with a shaded area representing the standard deviations and a continuous line representing the average values of the respective flow velocity components. It is visible that both the MD+ML as well as the ML-only approach deliver results matching the true particle data – except the aforementioned shift in u_y for ML-only, which is however clearly inside the standard deviation of the real MD data. Thus, this can be considered to be an excellent prediction result for a very noisy and challenging complex validation data set.

7 Conclusions

We have pointed out that a hybrid convolutional recurrent autoencoder is a promising approach to learn and predict molecular dynamics data in the context of a molecular-continuum coupled simulation, in order to explore potentials for acceleration of simulation execution. To make our results reproducible, we provided an implementation of the proposed approach based on PyTorch and documented the dataset generation and training processes. Unlike existing approaches, we have introduced a type of loss function for the RNN training that quantifies loss in the velocity space, instead of in the latent space, leading to an enhanced integration of the hybrid model components. We solved problems caused by interdependent flow components by introducing a triple model approach. Our results demonstrate that the hybrid CNN+RNN ML model does

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

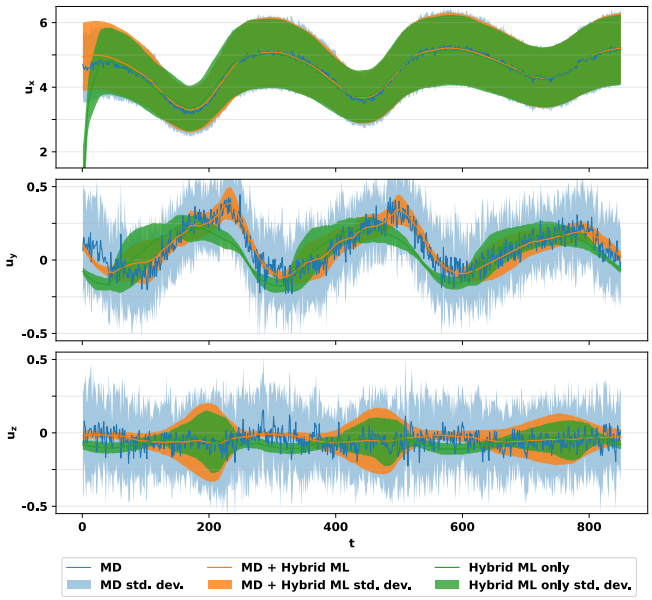


Fig. 8. Comparison of hybrid ML model performance with raw MD data in validation case 20000_NE, average and standard deviation of velocity for all cells in MD domain. Note that high ML output standard deviation here does not mean that the data is noisy, but that flow velocity varies over space for different cells, see also Fig. 1.

not only work as a powerful advanced noise filtering system, but even in a complex flow scenario, it is able to replace the MD system entirely and accurately predict the data that MD would generate for the coupled simulation.

We have considered one-way coupled simulations only. This was reasonable here because we wanted be able to analyze the behavior of the MD system and our ML model operating on particle data, or even analyze the ML module entirely replacing the particle system, independent of any additional effects which would stem from two-way coupling of the two systems. Thus, the impacts and limitations of the ML model in a two-way coupled setup constitute an open question for the future. While we have applied our approach to MD only, we expect it to naturally generalize to other types of particle systems, such as dissipative particle dynamics or smoothed particle hydrodynamics. They tend to expose the macroscopic solver to less noise and thus facilitate AE and RNN training, however their reduced computational cost makes it more difficult to obtain a significant performance benefit with a ML surrogate model. While we have shown that our ML model opens chances for drastic increases of computational efficiency, a further systematic investigation of the speed-ups achievable in practice by this method, including scaling to large-scale simulations on HPC clusters, would be essential. The fact that it can be expensive to re-initialise a MD simulation, after skipping some time steps, might play a crucial role here. So far the hybrid model was only tested for 'toy' scenarios where the average behavior of the MD system can be modeled by a macroscopic solver stand-alone.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

Thus, future work suggesting itself would be to conduct experiments with scenarios where the MD simulation actually provides differing physical characteristics, such as fluid-structure interaction or flows through carbon nanotube membranes.

Acknowledgments We thank the projects "hpc.bw" and "MaST" of dtec.bw – Digitalization and Technology Research Center of the Bundeswehr for providing computational resources, as the HPC cluster "HSUper" has been used to train and validate the ML models presented in this paper. We also want to thank Prof. Zhen Li and the MuthComp Group of Clemson University for fruitful discussions and exchange of ideas as well as provision of office space and IT systems.

Bibliography

[1] Bauer, M., Köstler, H., Rüde, U.: lbmpy: Automatic code generation for efficient parallel lattice boltzmann methods. Journal of Computational Science **49**, 101269 (2021)
[2] Borg, M.K., Lockerby, D.A., Ritos, K., Reese, J.M.: Multiscale simulation of water flow through laboratory-scale nanotube membranes. Journal of Membrane Science **567**, 115–126 (2018), ISSN 0376-7388
[3] Bungartz, H.J., Lindner, F., Gatzhammer, B., Mehl, M., Scheufele, K., Shukaev, A., Uekermann, B.: precice – a fully parallel library for multiphysics surface coupling. Computers I& Fluids **141**, 250–258 (2016)
[4] Fukami, K., Hasegawa, K., Nakamura, T., Morimoto, M., Fukagata, K.: Model order reduction with neural networks: Application to laminar and turbulent flows. SN Computer Science **2**, 1–16 (2021)
[5] Grinberg, L.: Proper orthogonal decomposition of atomistic flow simulations. Journal of Computational Physics **231**(16), 5542–5556 (2012)
[6] Jarmatz, P., Maurer, F., Wittenberg, H., Neumann, P.: Mamico: Nonlocal means and pod filtering with flexible data-flow for two-way coupled molecular-continuum hpc flow simulation. Journal of Computational Science **61**, 101617 (2022)
[7] Jarmatz, P., Neumann, P.: Mamico: Parallel noise reduction for multi-instance molecular-continuum flow simulation. In: International Conference on Computational Science, pp. 451–464, Springer (2019)
[8] Jarmatz, P., Wittenberg, H., Jafari, V., Das Sharma, A., Maurer, F., Wittmer, N., Neumann, P.: Mamico 2.0: An enhanced open-source framework for high-performance molecular-continuum flow simulation. SoftwareX **20**, 101251 (2022), ISSN 2352-7110
[9] Kadupitiya, J., Sun, F., Fox, G., Jadhao, V.: Machine learning surrogates for molecular dynamics simulations of soft materials. Journal of Computational Science **42**, 101107 (2020)
[10] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998), https://doi.org/10.1109/5.726791

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42

[11] Nakamura, T., Fukagata, K.: Robust training approach of neural networks for fluid flow state estimations. International Journal of Heat and Fluid Flow **96**, 108997 (2022)
[12] Nakamura, T., Fukami, K., Hasegawa, K., Nabae, Y., Fukagata, K.: Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. Physics of Fluids **33**(2), 025116 (2021), https://doi.org/10.1063/5.0039845
[13] Neumann, P., Bian, X.: Mamico: Transient multi-instance molecular-continuum flow simulation on supercomputers. Computer Physics Communications **220**, 390–402 (2017)
[14] Niethammer, C., Becker, S., Bernreuther, M., Buchholz, M., Eckhardt, W., Heinecke, A., Werth, S., Bungartz, H.J., Glass, C.W., Hasse, H., et al.: ls1 mardyn: The massively parallel molecular dynamics code for large systems. Journal of chemical theory and computation **10**(10), 4455–4464 (2014)
[15] Ren, X.G., Wang, Q., Xu, L.Y., Yang, W.J., Xu, X.H.: Hacpar: An efficient parallel multiscale framework for hybrid atomistic–continuum simulation at the micro-and nanoscale. Advances in Mechanical Engineering **9**(8) (2017)
[16] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI, pp. 234–241, Springer International Publishing (2015)
[17] Schäfer, M., Turek, S., Durst, F., Krause, E., Rannacher, R.: Benchmark computations of laminar flow around a cylinder. In: Flow simulation with high-performance computers II, pp. 547–566, Springer (1996)
[18] Smith, E.: On the coupling of molecular dynamics to continuum computational fluid dynamics. The school of Mechanical Engineering (2013)
[19] Tang, Y.H., Kudo, S., Bian, X., Li, Z., Karniadakis, G.E.: Multiscale universal interface: a concurrent framework for coupling heterogeneous solvers. Journal of Computational Physics **297**, 13–31 (2015)
[20] Thomas, M., Corry, B.: A computational assessment of the permeability and salt rejection of carbon nanotube membranes and their application to water desalination. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374**(2060), 20150020 (2016)
[21] Veen, L.E., Hoekstra, A.G.: Easing Multiscale Model Design and Coupling with MUSCLE 3. In: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (eds.) ICCS, pp. 425–438, Springer International Publishing (2020)
[22] Wiewel, S., Becher, M., Thuerey, N.: Latent space physics: Towards learning the temporal evolution of fluid flow. In: Computer graphics forum, vol. 38, pp. 71–82, Wiley Online Library (2019)
[23] Wittenberg, H., Neumann, P.: Transient two-way molecular-continuum coupling with openfoam and mamico: A sensitivity study. Computation **9**(12) (2021), ISSN 2079-3197, https://doi.org/10.3390/computation9120128
[24] Yin, W., Kann, K., Yu, M., Schütze, H.: Comparative study of CNN and RNN for natural language processing. arXiv.1702.01923 (2017)
[25] Zhang, A., Lipton, Z.C., Li, M., Smola, A.J.: Dive into deep learning. arXiv preprint arXiv:2106.11342 (2021)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36027-5_42