Machine learning and direct numerical simulation for loads prediction in particle suspensions

Written by:

Alberto Luigi Coronese

Intern Master 2 - Ingénieur Civil des Mines

01/04/2022 - 30/09/2022

Supervised by:

Thibault Faney

Gianluca Lavalle

Jean-Lou Pierson

IFP Energies Nouvelles

École Nationale Supérieure des Mines de Saint-Étienne

Politecnico di Torino

September 9, 2022

Contents

1 Introduction

This internship addresses multiscale modelling of dispersed phase flows. In particular, knowledge of the stresses to which solid particles immersed in a liquid flow are subjected is of paramount importance in order to better model the dynamics of two-phase flows.

This dynamic system is found in nature and in various industrial domains. Starting with water treatment, an increasingly topical issue related to safeguarding the planet and recycling raw materials, it is of particular importance to understand and manage polluting microplastic particles and fibers. Industrial processes such as fluidizer beds, bubble columns, and flotation processes for treating air pollutants require continuous scientific and engineering evolution.

Very often the continuous liquid phase and the dispersed solid phase are treated together through an Eulerian formalism. It first requires the mediation of transport equations and closure laws, including knowledge of the stresses on inclusions. The latter are of main importance for the precise prediction of pressure drops within certain industrial processes, as well as the prediction of their trajectories. Several models exist in the literature that can predict average stresses on solid particles, but very few models exist for local determination at the particle scale.

It is for the aforementioned purpose that we aim in this work to build an Artificial Neural Network model capable of giving us a prediction of the local efforts exerted on each inclusion. We first start with a simple benchmark model, and we then gradually improve upon this work using more DNS (Direct Numerical Simulation) training data, optimizing the model hyperparameter and transforming raw inputs into a symmetry preserving embedding.

This study is the first step towards a more in-depth analysis of the prediction model carried out within the IFPEN research centre, the company at which this internship was carried out. Additional information on it are included in Appendix A.

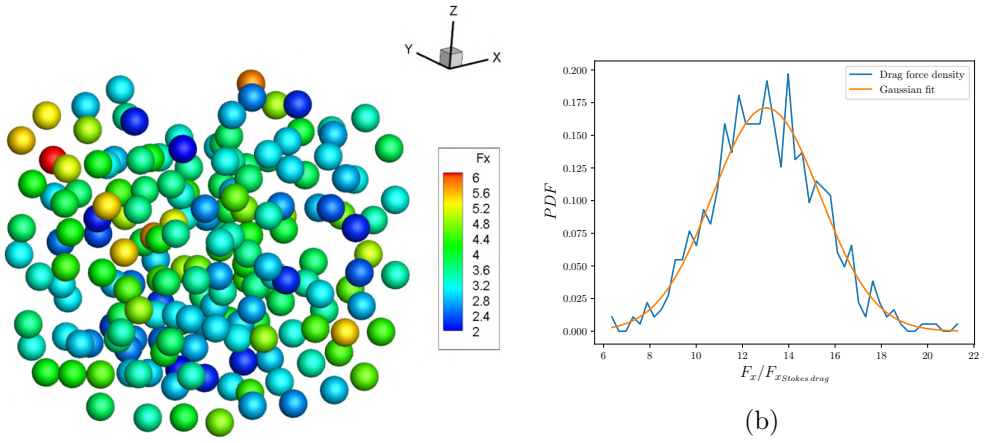
The programming language used is Python and the ML (Machine Learning) study was carried out using the Keras library integrated in Tensorflow [4].

2 Necessity of an ANN model

PR-DNS (Particle Resolved Direct Numerical Simulation) could be used to best model two-phase flows, but it has been found that the calculation time and computational cost are too high. Therefore, large-scale physical models are used to overcome this. They are composed of averaged formulations which, however, require closure laws. The empirical ones are very limited nowadays, because they require knowledge of the local stresses exerted on each inclusion in the domain.

The creation of a well-trained and optimised ANN (Artificial Neural Network) model could return us good predictions of these efforts, considerably shortening the calculation time. We learn from the literature that the distribution of stresses on the particles, in particular the resistance forces along the flow axis f_x, can deviate greatly from the average value.

Fig. 1a from the literature [1] shows us a colour-scale distribution of dimensionless f_x values on each particle resulting from PR-DNS. We note that dimensionless values range from 2 to 6, and looking at Fig. 1b taken from one of the simulations



(a) (b)

Figure 1: (a) Calculated drag force by PRS on an assembly of spherical particles at Re=10 and ϕ=0.1 [1] . (b) Histogram of f_x from a PRS at Reynolds number Re=10 and solid fraction ϕ=0.314 .

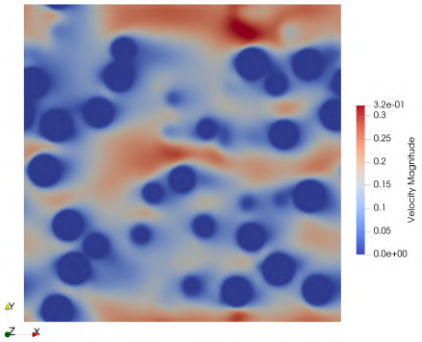
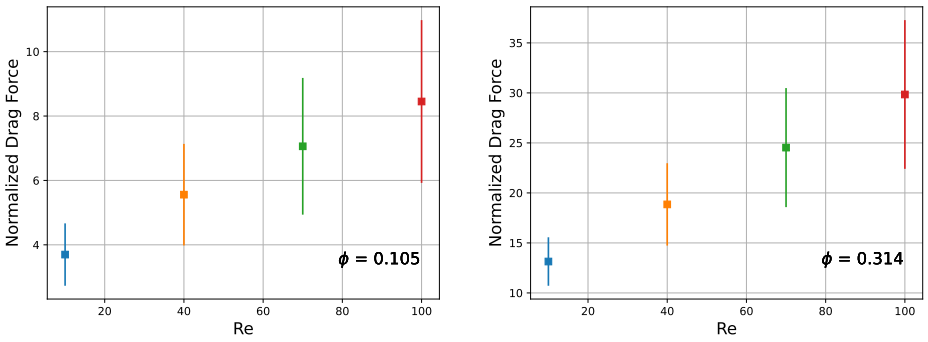


Figure 2: Velocity fluctuation distribution on y-plane at z = 0 for Reynolds number Re=10 and solid fraction ϕ=0.16

performed in this internship, we see that the f_x distribution follows a Gaussian law. The cause of this considerable difference between the values assumed by the f_x on the particles can be seen in Fig. 2, taken from the post-processing of one of the DNS simulations carried out in this work. The graph shows through the velocity distribution that each particle experiences a different flow depending on its position relative to neighboring particles, as it can be affected by wakes and flow perturbations. We now validate the importance of knowing the stresses on the particles instead of just the average stress.

We report in Fig. 3 the graphs of the dimensionless and averaged f_x as a function of the simulated Reynolds numbers for two solid fraction analyzed in the stage. The details of the CFD simulations carried out during the internship will be discussed in more detail later in the report; the information we retain from the above graphs are the remarkable values of the mean square deviation already from low Reynolds numbers, testifying to the wide range of stress values exerted on the particles.

Thus, all these analyses certify how important it is to go beyond the use of aver-



(a) (b)

Figure 3: Drag force f_x averaged with mean square error bars for Reynolds number Re=10, 40, 70, 100 with solid fraction (a) ϕ=0.105 and (b) ϕ=0.314.

aged f_x, as local values on inclusions would be overlooked. The ANN model would therefore be a good tool to integrate the empirical closure laws using the predictions of the stresses on each particle and be able to close the mediated formulations.

3 Starting parameters required for the model

For an effective prediction model, it is necessary to define which parameters of the problem most influence the value to be predicted. As this is a preliminary study, several simplifications were made with respect to the real multiphase flow system. The particles are modelled as spherical and with immobile position within an incompressible liquid in a stationary flow. In the model to be created, we only use as output the resistance forces f_x on the particles.

The physical parameters of the problem that are directly related to these predictions are: the flow velocity, taken into account through the Reynolds number Re,

$$Re = \frac{\rho\, U\, D}{\mu}, \tag{1}$$

where ρ, U and μ indicate the density, fluid velocity and dynamic viscosity of the fluid respectively, while D is the diameter of the spheres, taken as constant throughout the stage, and the solid volume fraction of particles dispersed in the flow ϕ,

$$\phi = \frac{4}{3}\, N\, \pi\, R^3, \tag{2}$$

where N is the number of spheres in the liquid volume and R is their radius.

As seen in Fig. 3, the geometrical parameters of the particles closest to the reference one also have their bearing on the prediction of the relative strength. We will start with a model that takes as input only the two physical parameters of the problem Re and ϕ, and then see how using information on the position of neighboring particles improves the prediction of f_x.

		Number of simulated particles			
N	Solid fraction ϕ	Re=10	Re=40	Re=70	Re=100
100	0.052	1200	1200	1200	1200
200	0.105	2400	2400	2400	2400
300	0.157	3600	3600	3600	3600
400	0.209	4800	4800	4800	4800
500	0.262	6000	6000	6000	6000
600	0.314	7200	7200	7200	7200
		Total database = 100800 particles			

Table 1: Summary of the quantity of DNS simulations performed by Reynolds number and solid fraction.

3.1 Configuration of DNS simulations

Before getting into machine learning models, it is a priority to have a good database through which future models can be trained and tested. All physical and geometric data and results are obtained from CFD simulations carried out in this first part of the internship. The tool used to perform simulations of fixed spherical particles dispersed in a liquid flow is PeliGRIFF, a parallel DEM-DLM/FD direct numerical simulation tool for 3D particulate flows.

The dimensionless Navier-Stokes equations for an incompressible flow are solved. Continuity:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3}$$

Momentum:

$$\frac{\partial u_i}{\partial t} + \frac{\partial (u_j u_i)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{Re}\left(\frac{\partial^2 u_i}{\partial x_j^2}\right) \tag{4}$$

The simulations are carried out using a constant particle diameter D of 0.1 mm and a cubic form of the reference liquid volume of dimension $10D$. The boundary conditions are tri-periodic and with a mesh counting 24 cells per diameter (240x240x240 computational background cells in the x, y and z directions, respectively).

3.2 Creation of a training database

To effectively train an ANN prediction model, it is crucial to have a large database of simulated particles at different velocities, solid fractions and geometric distributions.

For this reason, 12 particle distributions with random initial positions were simulated for a volume containing 100 to 600 spheres (solid fraction ϕ between approximately 0.05 and 0.32) and at Reynolds numbers Re between 10 and 100. The Tab. 1 contains details of the simulations and the total stored particles.

As mentioned above, the only results we retain from the simulations are the drag forces f_x acting on the particles along the flow axis alone.

They are first adimensioned using the Stokes-Einstein relation:

$$f_{x_i} = \frac{\tilde{f}_{x_i}}{F_{x_{Stokes\,drag}}} = \frac{\tilde{f}_{x_i}}{3\,\pi\,\mu\,D\,U} \ . \tag{5}$$

The average value for each solid fraction tested was calculated using the following formula:

$$\bar{F_x} = \frac{1}{N}\sum_{i=1}^{N} f_{x_i} \ . \tag{6}$$

However, some of the simulations did not achieve constant convergence, but for some particles the solution continued to undergo small periodic oscillations.

After verifying that the fluctuations were not due to a numerical problem, we agreed that indeed being simulations with high Reynolds number or low solid fraction, the problem was starting to become non-stationary. To avoid this, it is therefore decided to take the last simulation time steps (coinciding with the last oscillation periods) and to derive for each particle the average value of drag force exerted on it. This allowed us to avoid taking untrue f_x values and still use more data that would have proved useful for the model.

4 Benchmark

We now come to the creation of the first simplified ANN models, which we use as a starting point to build more advanced prediction models.

In this first modelling step, it is interesting to see the prediction efficiency of two basic models:

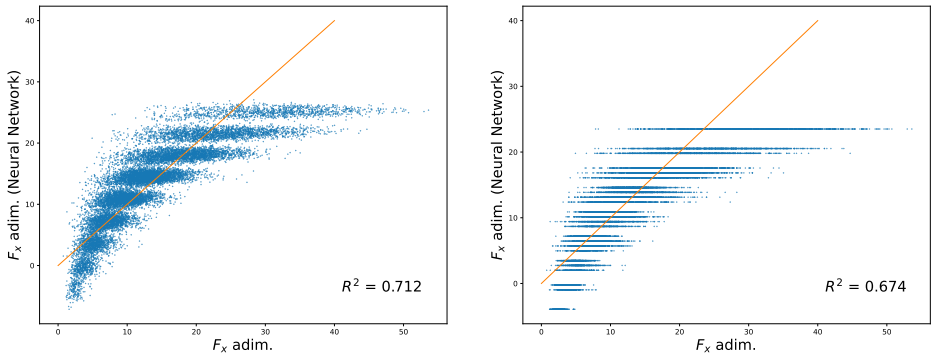
- the first using a simple linear regression (taking as input: Re, ϕ and the relative positions in cartesian coordinates of n neighboring particles);
- the second very similar to the model of Long He [1] in the hyperparameters set, but using only Re and ϕ as input.

In these benchmark models as well as in all more advanced models to be created, we use 60% of the accumulated data to train the model, and the remaining 40% is divided equally for the validation and testing phase. To assess the effectiveness of the models, we analyze the coefficient of determination R^2 for each ML simulation

$$R^2 = \frac{\sum_{i=1}^{N}(\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2} \ , \tag{7}$$

where $\hat{Y}_i$ are the predictions made by the ANN model, Y_i are the exact results derived from the DNS simulations and $\bar{Y}$ is the averaged drag force $\bar{F_x}$.

To go into the post-treatment graphs in more detail, we see in Fig. 4a the parity plot of the test phase inherent in the linear regression model. This specific scatter plot is particularly explanatory when prediction and machine learning studies are carried out, as it compares a set of results from a computational model (y-axis) with reference data (x-axis).



(a) (b)

Figure 4: Parity plots of the test phase referring to (a) the linear regression model and (b) the model using only Re and ϕ as input.

Principal model hyperparameters

#hidden layers	#neurons	activation func.	loss func.	optimizer	dropout
1	25	tanh	MSE	Nadam	0%

Table 2: Principal hyper-parameters used in the ANN model.

The linear regression model, using no activation function, has no learning potential, and its low efficiency is evidenced by the low R^2 value and the large difference visible graphically between the exact and predicted values of the resistance force.

In Fig. 4b, on the other hand, we see the parity plot of the test phase of a real ANN model (hyperparameters shown in the Tab. 2), but which does not take neighboring particles into account among the input parameters.

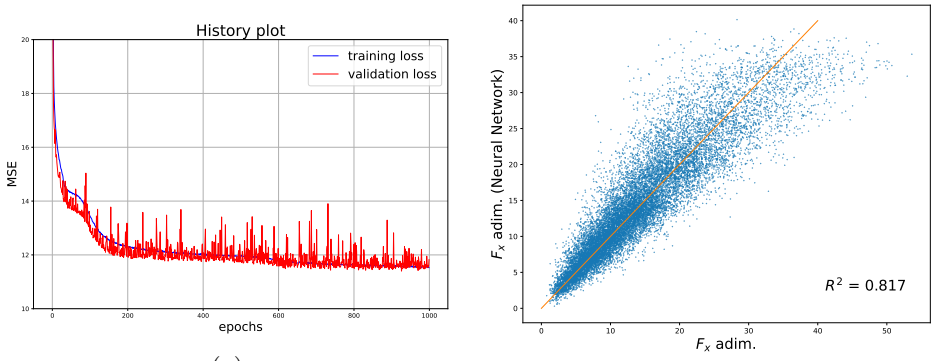
We see that the model actually only succeeds in distinguishing the predictions by (Re, ϕ) group, giving us only the average f_x between the particles. We note, therefore, that the model does not distinguish between f_x within the same simulation; this is a result we expected, as we did not give it as input any information at the level of the particle to enable it to make a local prediction.

5 Literature-based modelling

Once we were familiar with the simpler models mentioned above, we created a first optimal ANN model for predicting the f_x exerted on each particle. We used the previously defined hyperparameters and largely derived from the model in the literature by Long He [1]. The only differences, apart from the database used, are:

- in the activation function, which remains very similar to the tangent sigmoid used in literature [1];
- in the optimisation method used to minimize the training loss, which has a negligible influence on the final prediction result, but rather plays a fundamental role on the learning rate of the ML simulation.

Let us now focus on defining the inputs that make the prediction different from that presented by the second model in the 4.



(a) (b)

Figure 5: (a) History plot of the loss function for the training and validation phase. (b) Parity plot of the test phase of the model based on literature [1].

Among the inputs along with Re and ϕ were added the relative positions in Cartesian coordinates of the 15 particles placed closest to the particle relative to the prediction; thus for this particular model we have a total of no less than 47 elements in the input vector. To follow the training of our model and thus its optimisation, we visualise how the loss function varies as a function of the epochs, i.e. the simulation steps. Throughout the work, we used the MSE (Mean Squared Error)

$$MSE = (\hat{Y}_i - Y_i)^2 \ , \tag{8}$$

where $\hat{Y}_i$ are the predictions made by the ANN model, Y_i are the exact results derived from the DNS simulations. The graph in question can be seen in Fig. 5a, where the two simulation phases, training and validation, are clearly distinguishable.

In the parity plot in Fig. 5b, on the other hand, we can already see a considerable evolution in the predictions of our model compared to those of the benchmark, due to the neighbourhood taken into account in the inputs.

Since we use the same prediction model, the graph allowing us to compare our results with those in the literature is the one in Fig. 6, i.e. the relative error of the predictions as a function of the cumulative distribution. In other words, this graph shows us the number of particles as a percentage whose f_x prediction falls within a given relative error of the exact results. The formulas used to calculate this error, for predictions and mean values, are as follows:

$$error_{ANN} = \frac{F_x^{ANN}(Re,\phi,i) - F_x^{PRS}(Re,\phi,i)}{F_x^{PRS-MEAN}(Re,\phi)} \ , \tag{9}$$

$$error_{mean\,value} = \frac{F_x^{PRS}(Re,\phi,i) - F_x^{PRS-MEAN}(Re,\phi,i)}{F_x^{PRS-MEAN}(Re,\phi)} \ , \tag{10}$$

where F_x^{ANN} are the ANN predictions, F_x^{PRS} are the exact F_x and $F_x^{PRS-MEAN}$ are the averaged $\bar{F}_x$.

In the graph, we see how our model and that of the literature are both more effective than using only the mean value of f_x for each particle, but we also see how our model returns less accurate prediction results. In addition to the already

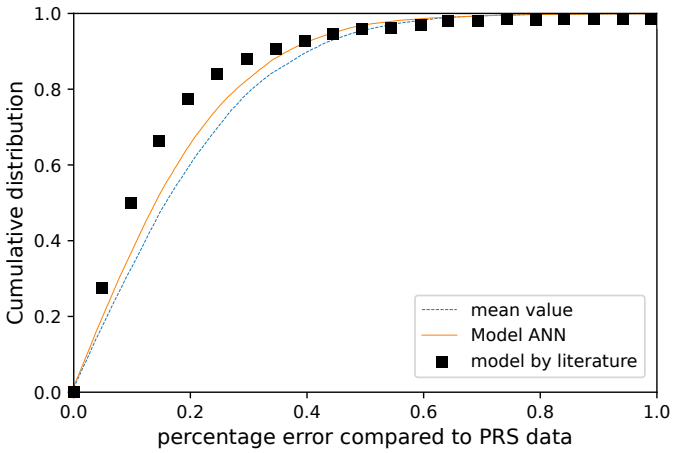


Figure 6: Cumulative distribution function of relative error over all Reynolds numbers and solid fractions among using the ANN predictions, the mean value of f_x and the literature results [1].

discussed negligible differences in the model settings, the real difference between our results and those of the literature is the use of a completely different database, which is specific to the DNS simulations performed. Conscious of this, we have steered the work towards an optimisation of the model by making several modifications. First, we see in the next chapter the standardisation of the inputs and the modification of the data splits for the three simulation phases of the model. A more comprehensive set of graphs inherent to the model just seen can be found in Appendix B.

6 Model optimisation

6.1 Standardisation of inputs and modification of splits

The first modification made to our basic model presented in the previous chapter was the standardisation of the inputs. In more detail, when analysing the magnitudes of the different inputs, we see a heterogeneity due to the different origin, be it physical or geometric. In effect, the model finds itself having to evaluate in the same learning step values from 10 to 100 for the Reynolds number, from 0.05 to 0.32 for the solid fraction and all the different relative positions of neighboring particles.

To standardise these elements on the same scale, we used the $Scikit-learn$ [2] package on python, and standardised the variables by removing the mean and scaling to unit variance. On the other hand, as far as the outputs are concerned, since they are only the f_x, we did not consider it necessary to make any changes, thus leaving them simply scaled with Stokes Drag. What emerges is that the model is slightly more effective simply with this standardisation of inputs; for this reason, we will only use standardised inputs from now on.

We then performed a test to see how much the division of the data between the training, validation and testing phases affected the accuracy of the predictions. In particular, the split made so far involved an orderly division within each simulation for each phase, i.e. the first 60% of particles went to the test phase, the next 20%

Range of tested hyperparameters

#hidden layers	#neurons	activation function	optimizer	dropout
[1, 50]	[1, 50]	tanh	RMSprop	[0, 50%]
		sigmoid	SGD	
		softma	Adam	
		relu	Nadam	

Table 3: List of hyperparameters tested during the optimisation performed on *Optuna*.

to the validation phase and the remaining 20% to the test phase. Now we will no longer divide the particles within the same simulation, but rather dedicate the total particles of 60% of the simulations performed per pair of Re and ϕ to the test phase and so on. This ensures that each phase will test particles from separate simulations and not different particles from the same simulation. In addition, the choice of simulations dedicated to each ML simulation phase is made randomly from those performed per pair of Re and ϕ.

6.2 Optimisation of hyperparameters using Optuna

The actual optimisation of the ANN model was carried out through the automatic hyperparameter optimisation software framework *Optuna* [3], an imperative define-by-run style user API. It enjoys high modularity, and with that we can dynamically construct the search spaces for the hyperparameters.

In addition, Optuna uses a Buleian method of optimisation research. Through various simulations and hyperparameter values, it selects the most effective combinations of these to optimise the quantity taken as a reference. In our case, this coincides with the already described coefficient of determination R^2.

The hyperparameters that were tested with their respective ranges of values or types are presented in the Tab. 3.

A very useful tool for post-processing visualisation of model results is the TensorBoard suite of web applications, a visualisation tool provided with TensorFlow. After a brief analysis of the trade-off between model complexity and model efficiency, we arrived at the definition of the new optimised Keras hyperparameters:

- 2 hidden layers;
- 40 neurons per layer;
- activation function: tanh;
- optimizer: Nadam;
- dropout: 4%.

6.3 Study of neighboring particles

After defining the main hyperparameters whose combination creates the most efficient model, we tested the latter by giving it a variable number of neighboring particles as input data to see how the accuracy of the predictions varies. Therefore,

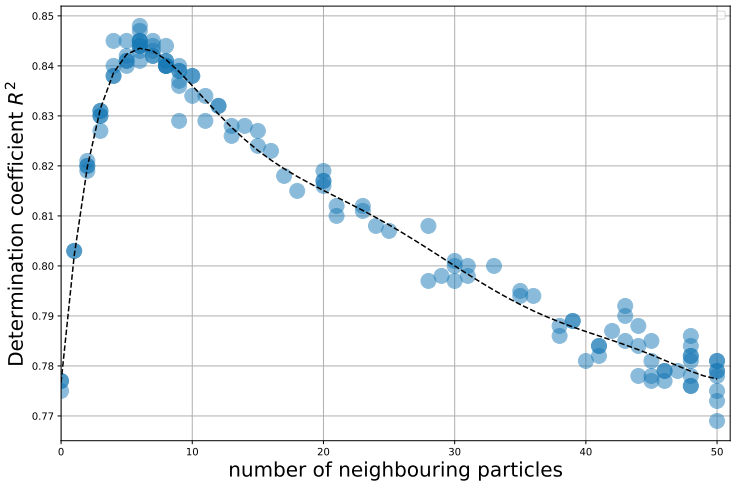


Figure 7: Coefficient of determination of the ANN model as a function of the number of neighboring particles taken as input.

for this purpose, we created the graph in Fig. 7, where we see as a function of the neighboring particles taken into account the values of the coefficient of determination derived for each case tested.

We note that the optimal value of neighboring particles that optimise the predictions is about 6, a lower number than we expected when comparing with the literature [1]. In fact, in the latter, the model's predictions improve from 5 to 15 particles, whereas our model seems to be unable to optimally handle the high number of relative coordinates of the input particles. So we are in the condition where our model is more efficient with a low number of neighbours, but not having enough information it is less accurate in its predictions. When we increase the number of neighbours, however, the model seems to overfit the training data set.

6.4 Prediction results

Using the optimised model with input: Re, ϕ and the relative cartesian coordinates of the 6 nearest particles, we present in Fig. 8 the comparison with the other variants of the model.

Comparing the relative error of the cumulative particle distribution of the basic model and the optimised model (both with standardised inputs), we see that the curves are practically all overlapping. This makes us realise that the model used in the literature, although relatively simple in terms of hidden layers and neurons used, is already optimised for our data and prediction system.

In addition, in this graph we present the error distributions resulting from the two ANN models tested so far for both types of data splitting, the ordered intra-simulation and the random one for different simulations on the three ML simulation phases. For both models, we see that the type of split does not play a decisive role on the efficiency of the model. Further graphics produced for the post-processing of the model can be found in Appendix C.

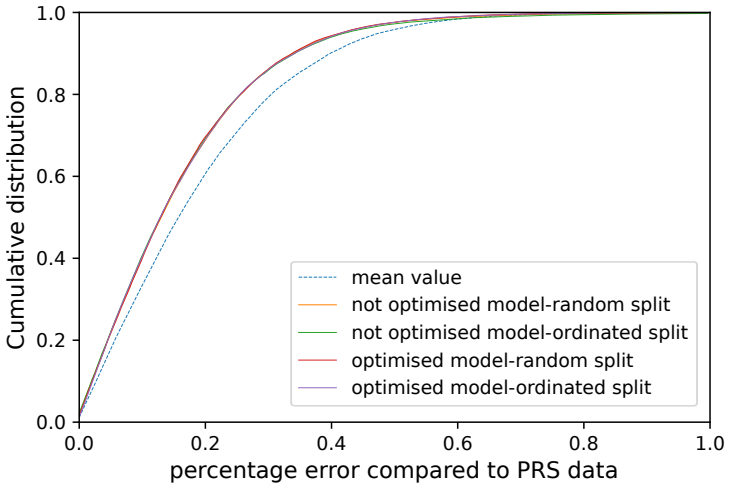


Figure 8: Cumulative distribution function of relative error over all Reynolds numbers and solid fractions among using the original model, the one optimised and the mean value of f_x.

7 Switching to the polar coordinate system

Observing the last model and its difficulty in handling the numerous inputs related to the neighborhood geometry of each particle, we carried out an input optimisation study. More specifically, with reference to the diagram in Fig. 9, we have noticed

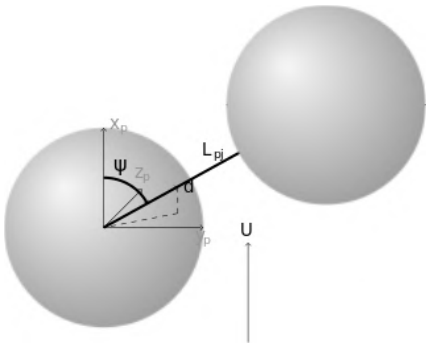


Figure 9: Representative diagram of two neighboring particles and polar geometric quantities.

that each spherical inclusion influences axisymmetrically the particles close to it according to the direction of flow. Therefore, in addition to the Re and ϕ always taken into account, any information other than the relative distance between the two particles and the angle formed with respect to the direction of flow $_p$ is to be considered superfluous.

The relative distance d takes into account the importance of affection between the particles and the ψ angle the relative position per pair of inclusions. In this way we have one less input per neighboring particle taken into account and one less piece of information that is truly useful to the model.

Once we had resized our database with the new polar coordinates, we immediately tested it on the model we had optimised for cartesian coordinates. In doing so,

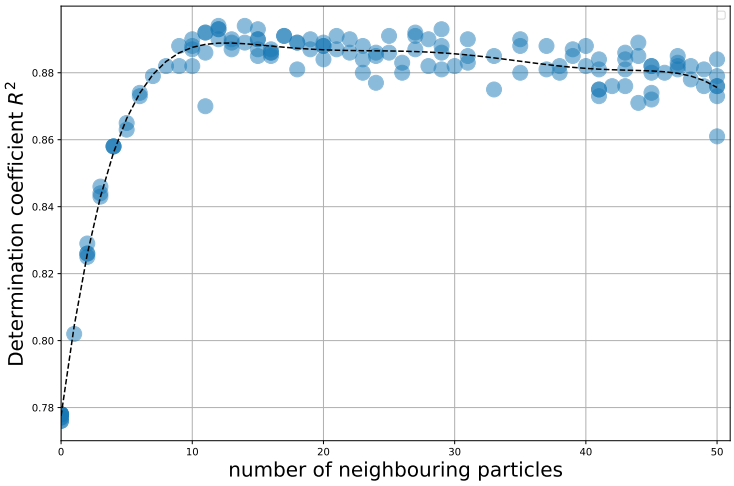


Figure 10: Coefficient of determination of the ANN model with polar coordinates as a function of the number of neighboring particles taken as input.

we have a first prediction model that we go on to optimise for the new coordinates again using *Optuna*. To analyse whether optimisation will play a decisive role on predictions this time, we will plot the relative error curve for the two models.

The optimised Keras hyperparameters of the ANN model with these new inputs are as follows:

- 6 hidden layers;
- 18 neurons per layer;
- activation function: relu;
- optimizer: RMSProp;
- dropout: 0%.

7.1 Study of neighboring particles

As done previously with cartesian coordinates, we test the new optimised model with a variable number of neighboring particles taken into account in the inputs.

Through the graph in Fig. 10, we see how the accuracy of the predictions varies by plotting the coefficient of determination R^2 for each case tested. Indeed, by using polar coordinates, the model seems to be able to handle a larger neighborhood more robustly, and thus make efficient use of more information. From 14 particles onwards, the prediction accuracy decreases very slightly, but even with 50 particles the model still performs better than the highest R^2 values achieved so far.

7.2 Prediction results

In Fig. 11 we see the parity plot of the test phase, which indeed shows us that the predictions are far more accurate than those obtained so far with the previous models, reaching a coefficient of determination R^2 equal to 0.889.

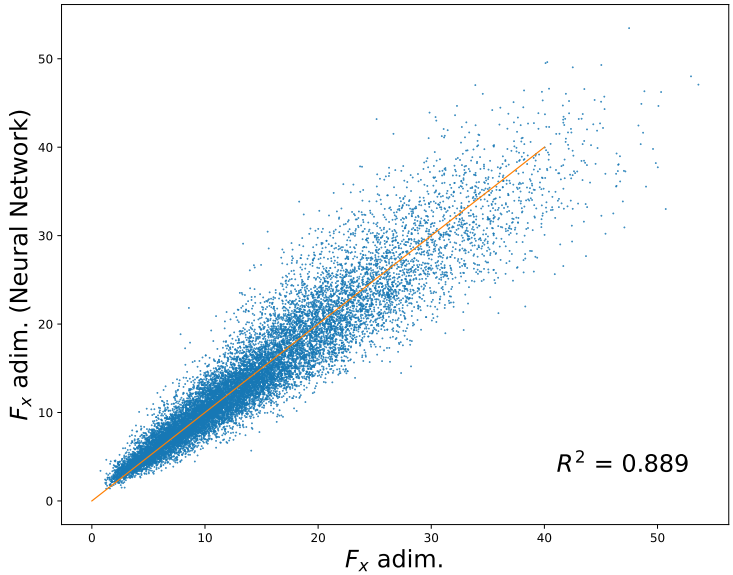


Figure 11: Parity plot of the simulation test phase of the optimised ANN model with polar coordinates.

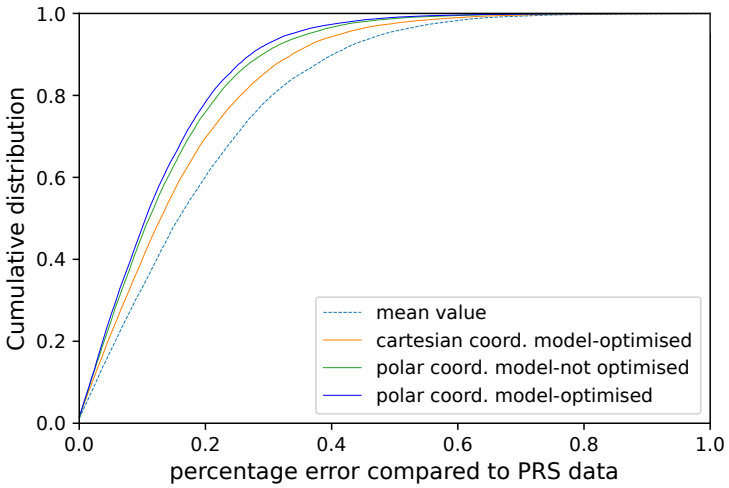
Let us now compare in the error graph in Fig. 12a the new model with the previous one using cartesian or polar coordinates. We see that already the results obtained with the non-optimised model are better than all those tested with cartesian coordinates, testifying to the real value added by changing the inputs. Furthermore, in this case, the optimisation performed on *Optuna* give us a good fit of hyperparameters that increase the accuracy of the predictions.

The significance of this improvement can be seen in the latest comparison with the literature in Fig. 12b. Here we see that our model's predictions are even more accurate than those presented by Long He [1], especially for those particles subject to drag forces more extreme of our domain.

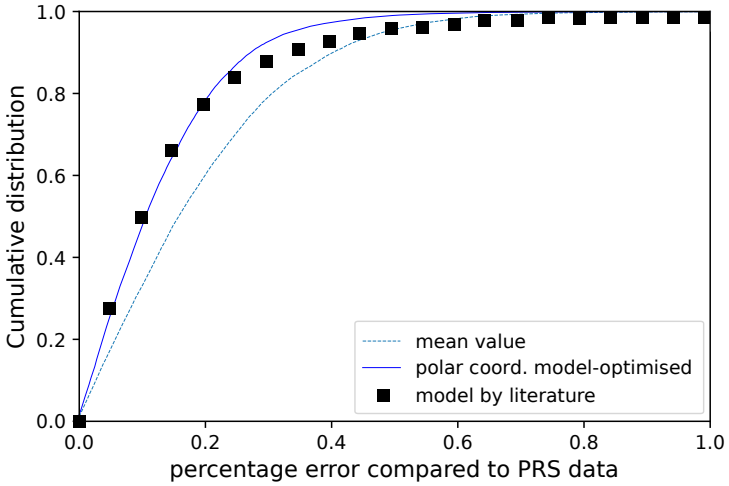
In conclusion, we show in Fig. 13 the comparison for each simulation macro group (Re, ϕ) for the 3 main models presented:

- the model based on the hyperparameters in the literature [1] (neighbourhood in cartesian coordinates);
- the model with hyperparameters optimised with *Optuna* (neighbourhood in cartesian coordinates);
- the final model optimised for inputs taken in polar coordinates.

These graphs show for each particle (indeed in ascending order with respect to the exact f_x exerted on it) the drag force calculated by means of the DNS-PRS simulations, the prediction of the same force made by means of the reference ANN model and the average prediction value calculated for each 20 particles. First of all, we note how the f_x derived from the simulations deviates considerably from the average f_x over all inclusions, graphed through the dashed horizontal line.



(a)



(b)

Figure 12: Cumulative distribution function of the relative error over all Reynolds numbers and solid fractions of the results of (a) the last three models described and (b) our last model and that of the literature [1].

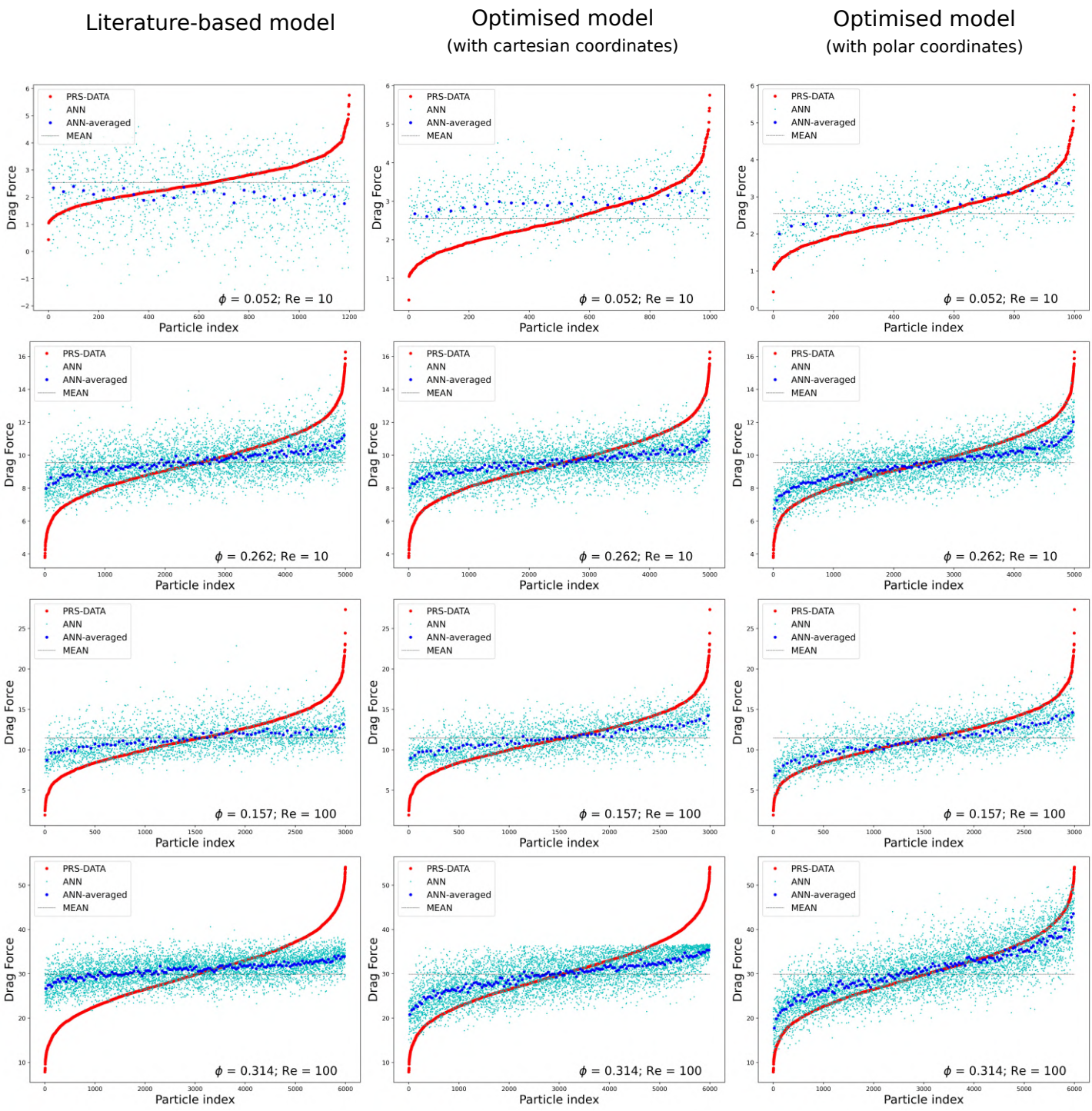


Figure 13: ANN results compared to PRS data, mean value at corresponding Re and ϕ, and averaged ANN value.

Given this, our aim is to get the blue distribution of predictions as close as possible to the red distribution of correct f_x. We note that for the graphs referring to low solid fractions, the number of particles tested is significantly lower. Logically, this is due to the fact that, for the same number of simulations and the same fluid volume, we have fewer inclusions and thus a lower solid fraction. As witnessed also by the comparisons carried out previously, the latest ANN model (right side) provides us with more accurate predictions than the other two models, which are very similar in their results. In fact, the most significant improvement is seen for the most extreme values of the f_x, where the model manages to be more effective.

A complement of graphs is presented in Appendix D.

8 Conclusions and perspectives

The aim of this stage is to create an Artificial Neural Network model capable of predicting as accurately as possible the stresses exerted on individual particles dispersed in a liquid flow. By limiting ourselves to the drag force along the flow axis and simplifying the particles to the spherical shape, we first assembled a good database through DNS simulations. After defining our inputs and an initial model based on the literature, we obtained results that were less precise than those we refer to. Thus, with the aim of improving the model, we made several changes, both in the data used to train the model and by optimising its hyperparameters.

The role of data division, input standardisation and the number of inputs are also studied. While the division of data plays a negligible role on the efficiency of the model, the standardisation of inputs is a modification we have retained for further development. Although we have tested numerous hyperparameters to improve the predictions, the apparently optimised model still performs at the same level as the base model. We assume that this difference in accuracy is mainly due to the data used to train the model, among which a good portion of f_{xi} was subjected to preprocess of averaging, due to the instability of some simulations at low ϕ and high Re.

By performing a relative coordinate analysis, we simplified the information to be given to the model in the inputs, replacing coordinates in the cartesian system with polar ones. Once the model has been re-optimised for this new input configuration, the improvements obtained are significant, achieving even slightly better accuracy than that obtained in the literature. So on the basis of the data in our possession and an initial development of a machine learning model, we arrive at an acceptable optimisation of the predictions, arriving at the definition of additional considerations compared to the studies presented in the current literature.

Possible further steps are firstly to use the created model to predict other efforts currently already stored by the PR-DNS, such as resistance forces along the y and z axes and torques to analyse the degree of accuracy.

An additional complexity is the change in the shape of the inclusions. In reality, these can often be assimilated to a rather cylindrical shape, but this would entail additional information to be taken into account in the model, i.e. the relative orientation between the particles.

A final interesting development that could be extremely useful is the definition of an analytical formulation capable of returning the predicted values on individual inclusions. To do this, the function should take as input a set of variables similar to those used in our study, starting of course with the Reynolds number and the solid fraction to obtain an initial value averaged over the set of particles. Instead, the most precise calculation would come with the enrichment of the formulation through terms based on the pairwise additivity hypothesis, which take into account the relative distances and sizes of inclusions. In this case, the creation of an ANN model would be necessary in defining the coefficients and constants to be included in the formulation.

References

[1] Long He, Danesh K. Tafti, A supervised machine learning approach for predicting variable drag forces on spherical particles in suspension. *Powder Technology 345* (2019), 379–389.

[2] Pedregosa *et al.*, Scikit-learn: Machine Learning in Python. *JMLR 12* (2011), 2825-2830.

[3] Akiba, Takuya and Sano, Shotaro and Yanase, Toshihiko and Ohta, Takeru and Koyama, Masanori *et al.*, Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019).

[4] Martín A. *et al.*, TensorFlow: A System for Large-Scale Machine Learning. Software available from tensorflow.org (2015).

A Appendix : IFPEN company description

IFP Energies Nouvelles, originally in the oil and gas industry, is a player actor in different research domains like circular economy (Plastic recycling, Environmental monitoring, CO2 capture, etc.), climate and environment, renewable energies (Biofuels, Biogas, Biochemistry or Green chemistry, Hydrogen, etc.) and sustainable mobility (Hybrid and electric powertrains, Connected vehicles, etc.).

The company has around 1600 employees, on two sites (Rueil Malmaxison near Paris, and Solaize near Lyon). Among them, 70IFPEN contains 9 different research and innovation divisions (Analysis, Catalysis biocatalysis and separation, Mobility and systems, Numerical science, Process design and modelling, Physico-Chemistry and Applied Mechanics, Process experimentation, Technology watch and economy, Earth science and environment).

This internship takes place at the Physico-Chimie et Mécanique Direction / Dpt. Mécanique des Fluides division (R174) and it is supervised by Thibault FANEY (R115 department) and Jean-Lou PIERSON (R174 department), both research engineers at IFPEN.

B Appendix : First literature-based model

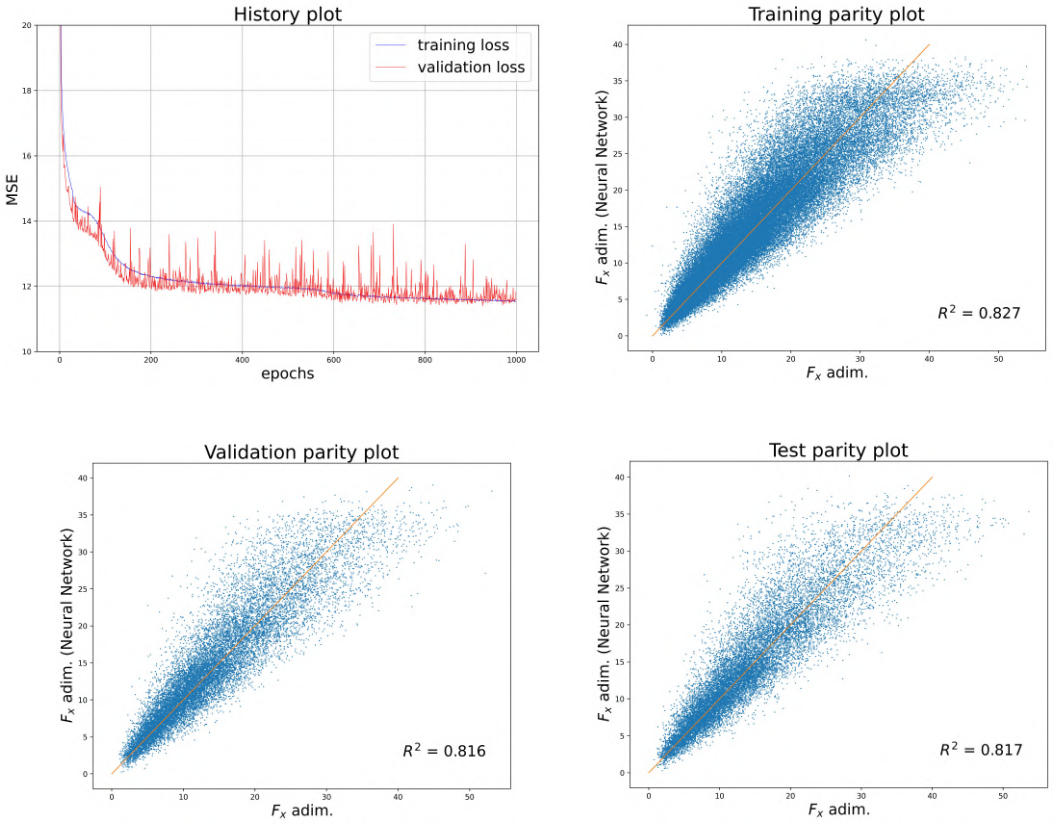


Figure 14: History plot of the Mean Square Error of the training and validation phase as a function of epochs. Parity plots of the Training, Validation and Test phases.

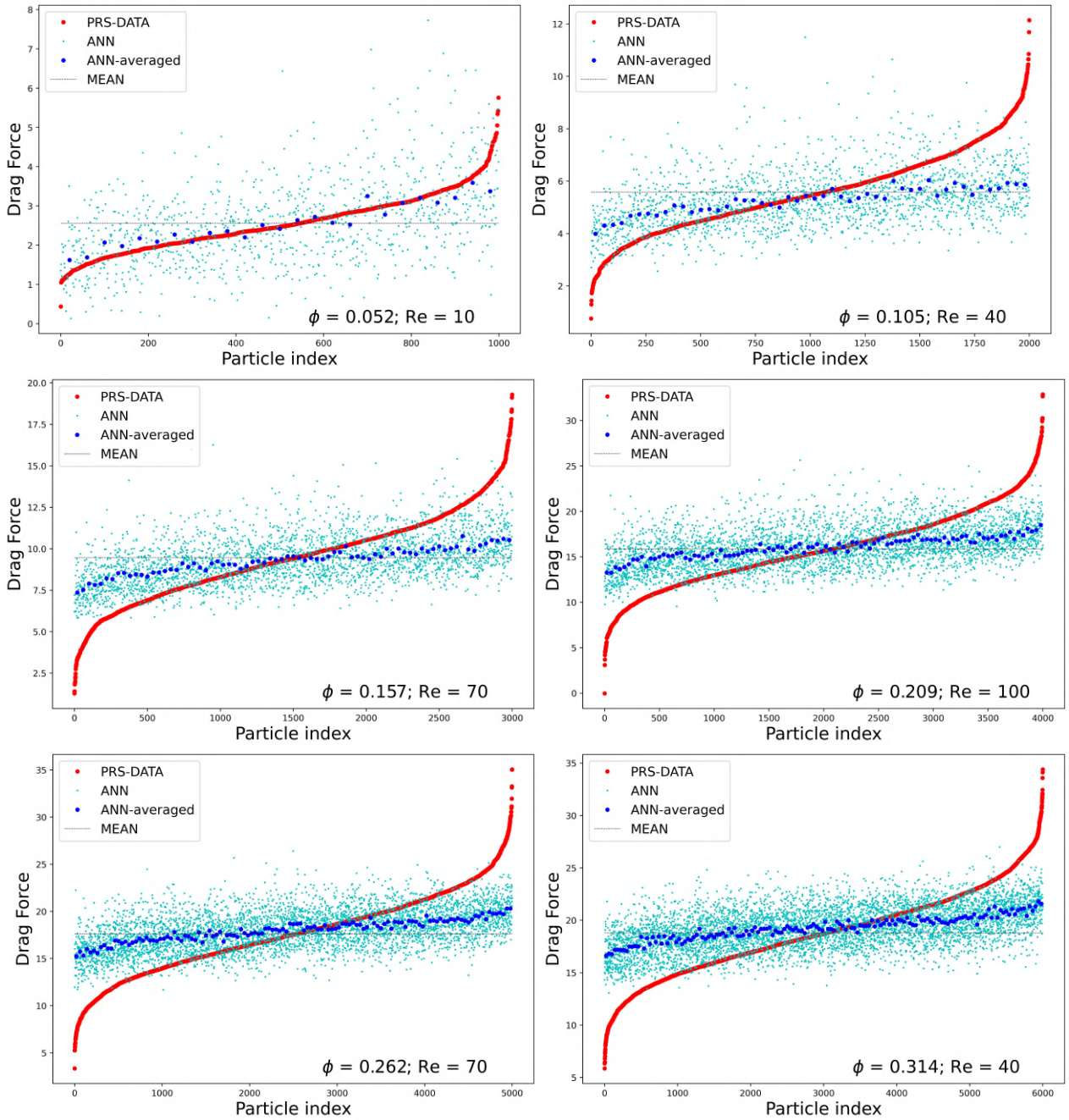


Figure 15: ANN results compared to PRS data, mean value at corresponding Re and ϕ, and averaged ANN value.

C Appendix : Optimised model

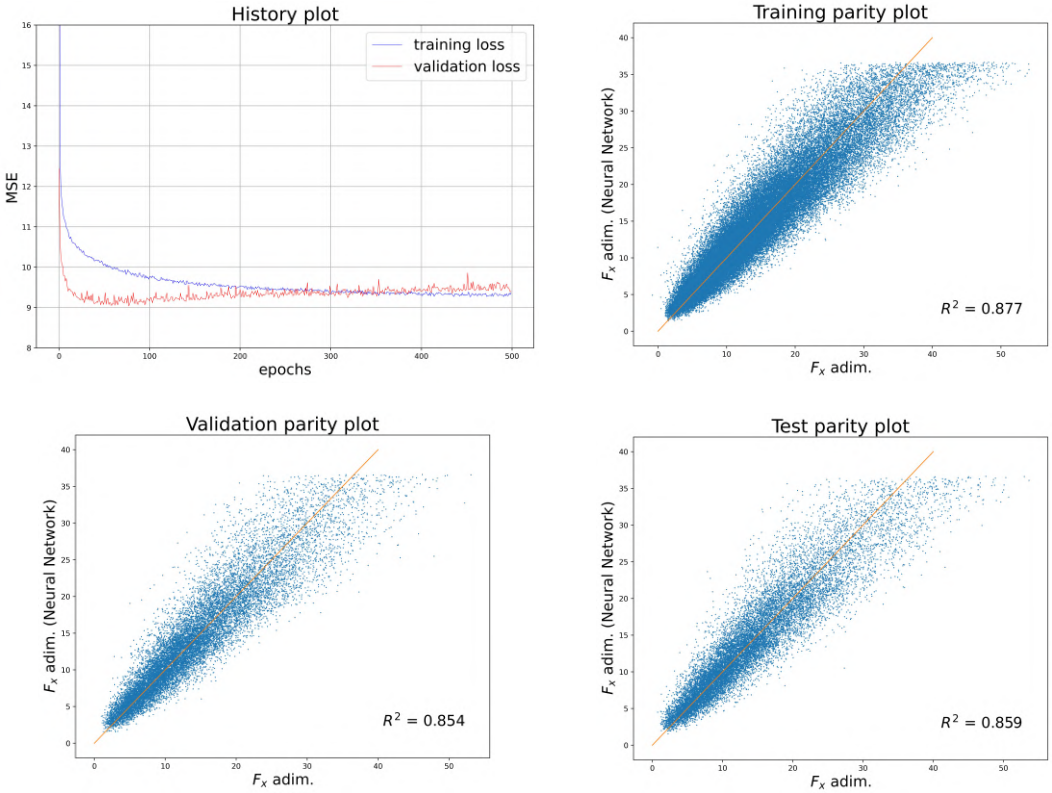


Figure 16: History plot of the Mean Square Error of the training and validation phase as a function of epochs. Parity plots of the Training, Validation and Test phases.

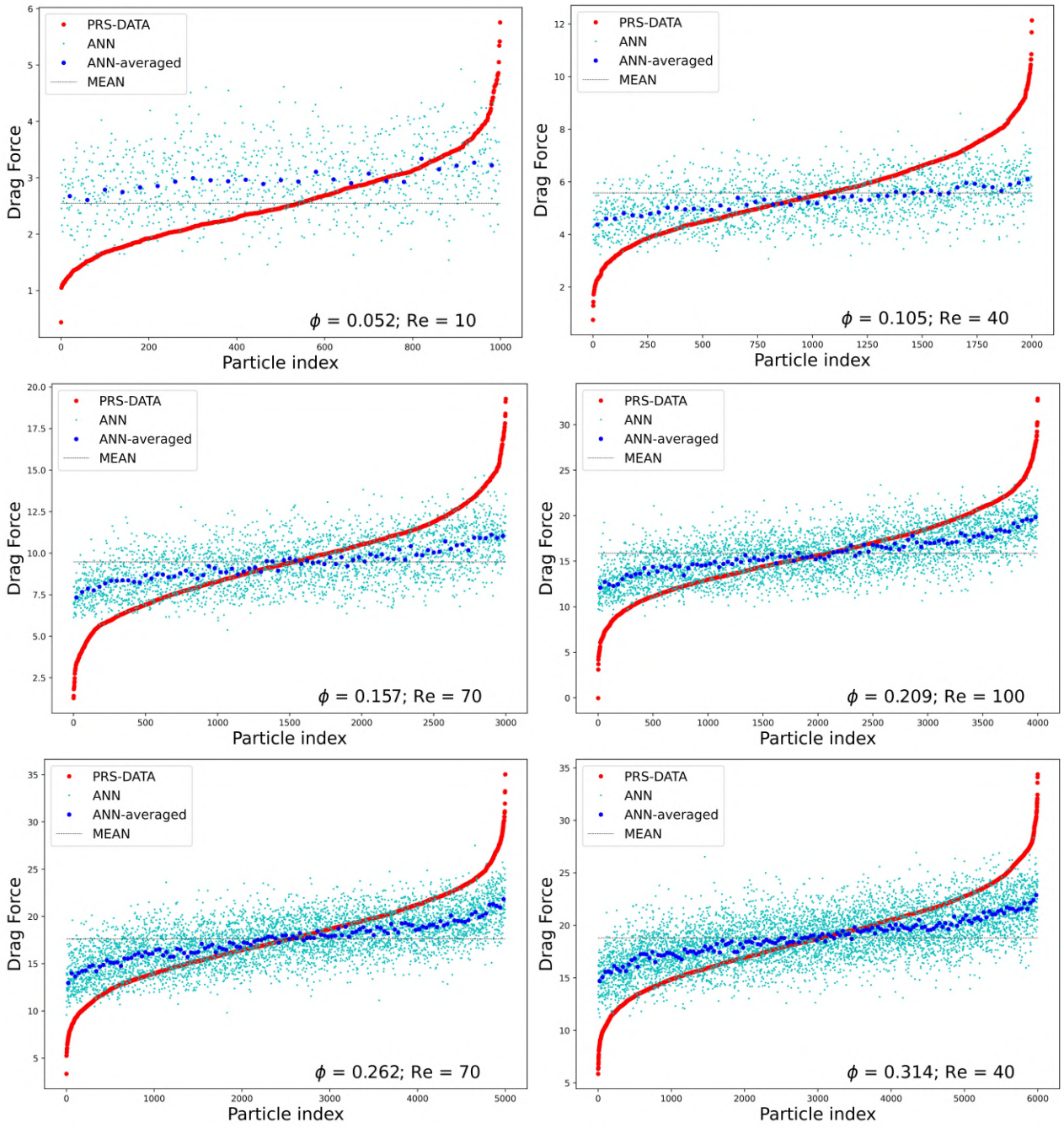


Figure 17: ANN results compared to PRS data, mean value at corresponding Re and ϕ, and averaged ANN value.

D Appendix : Final model with polar coordinates

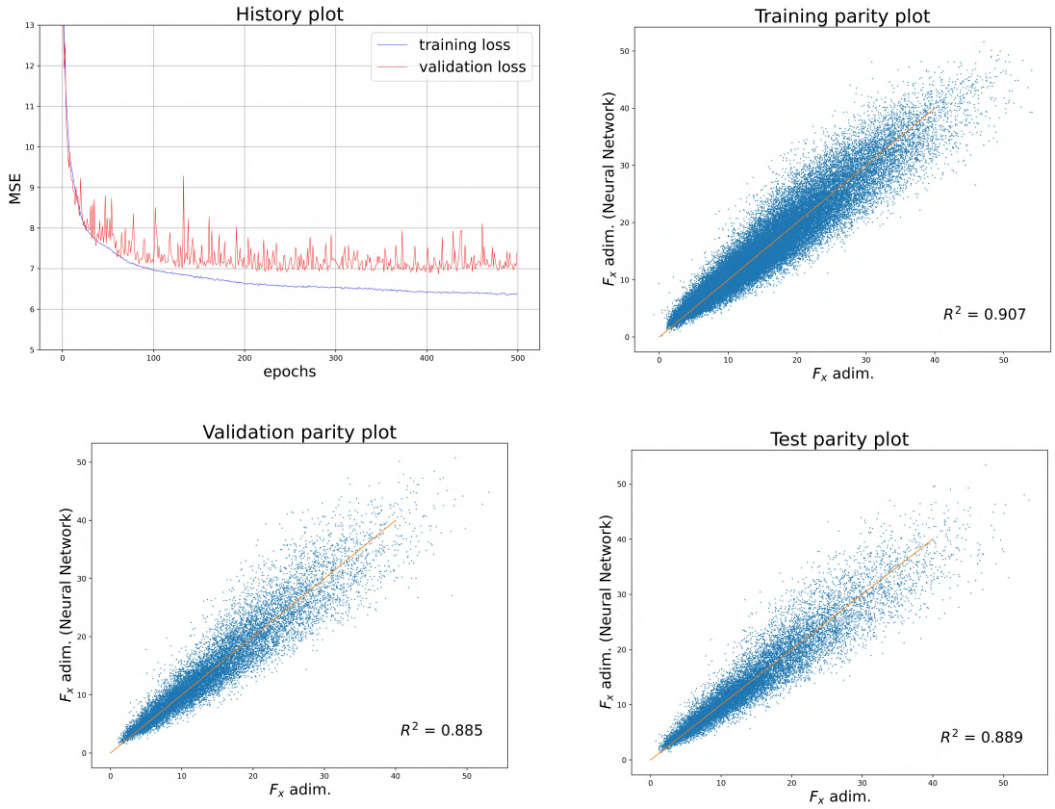


Figure 18: History plot of the Mean Square Error of the training and validation phase as a function of epochs. Parity plots of the Training, Validation and Test phases.

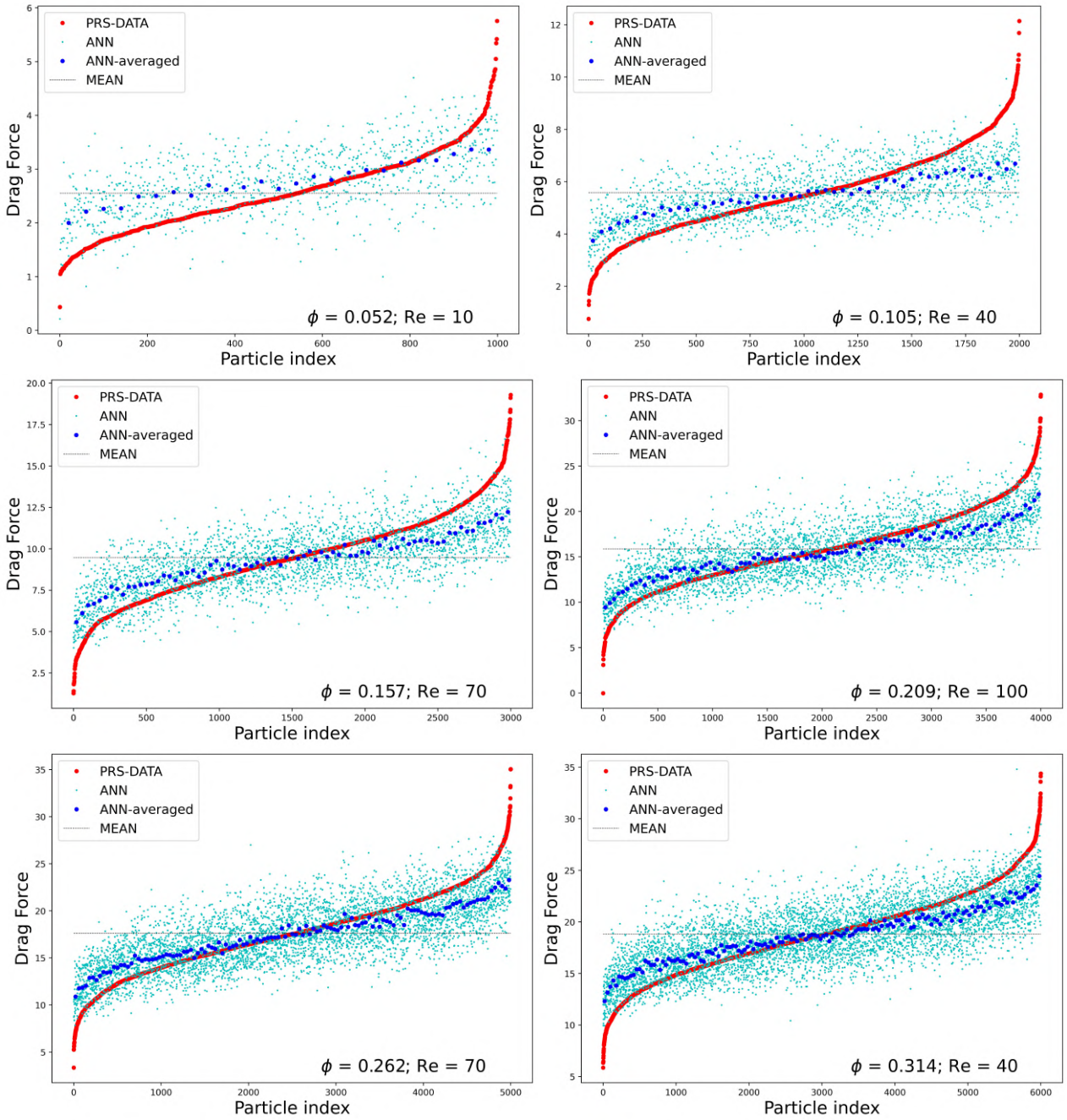


Figure 19: ANN results compared to PRS data, mean value at corresponding Re and ϕ, and averaged ANN value.