DOI: 10.15480/882.9314

Eighth International Symposium on Marine Propulsors
smp'24, Berlin, Germany, March 2024

# Neural Networks based Processing of CFD Data for supporting the Early Design of Ship Propellers

**Maike Strecker**[1], **Martin Scharf**[1], **Moustafa Abdel-Maksoud**[1]

[1]Institute for Fluid Dynamics and Ship Theory, Hamburg University of Technology, Hamburg, Germany

**ABSTRACT**

While powerful and reliable CFD methods are available today, designing propellers using them is unfortunately time-consuming and cost-intensive. On the contrary, existing CFD results for the hydrodynamic behaviour of propellers can be stored and processed with neural networks. This paper deals with the acceleration of the early propeller design stage by using this data for the training of neural networks.

For the generation of the dataset for the network training different radial propeller parameters including skew, chord length, camber and pitch for three to six bladed propellers were developed. The outputs of the automatic CFD calculations are the pressure distribution over the blade surface and the thrust and moment coefficients for different advance ratios.

Then all parameters are processed in a two-step neural network; the first neural network processes the operating point data and the second network the geometry data and estimated cavitation area. Based on the requirements such as installed power or the required propeller thrust the optimal geometry regarding a maximized efficiency is determined.

The grid sensitivity study in combination with processing different amounts of training data show that with sufficient data quality, a higher amount of data reduces the averaged error values in both networks until a certain point.

**Keywords**

propeller design, neural networks, computational fluid dynamics

## 1 INTRODUCTION

Especially in the early design phase the propeller design should be conducted in a limited time span while already being fairly precise. RANS methods are very powerful today, but they are time-consuming. The idea followed in the present study is to use existing data from CFD calculations to train neural networks to be able to speed up the design process in the early design phase.

## 2 RELATED WORKS

As mentioned before, the design process of a propeller is computationally expensive so that alternatives need to be developed. Vardhan et al. (2021) trained a random forest tree regression using certain input variables to design a propeller. As input the vessel's speed, propeller rpm and necessary thrust are used and the output are the geometric parameters such as number of blades, propeller diameter and radial pitch and chord distributions. The training data is generated by applying the OpenProp method. However, cavitation as an important parameter for propeller design is not considered. This is going to be assessed within this research.

A different approach with a similar result was considered by Gypa et al. (2021). They explored a combination of interactive genetic algorithms and machine learning, precisely the tool support vector machine (SVM). They suggested to use interactive genetic algorithms for propeller design variations which were then evaluated by an SVM. The SVM was trained on the first populations, where human experts assessed the individuals regarding their performance and cavitation area by selecting the next generations parents. The approach provided fairly good results compared to a single assessment by human experts. Following this procedure, the time which human experts spend on the design process can be reduced, while cavitation is considered. However, it still does not exclude some time-intensive computational processes during the design phase.

In a more recent work, Gypa et al. (2023) reevaluated their interactive optimization approach considering pressure and suction side cavitation on two more complex designs. Furthermore, they compared several machine learning algorithms, for example SVM as before, decision trees and also neural networks for the classification of the design within the generations. For one propeller design the neural network performed more efficiently, for the second the SVM. The results of this design process are comparable to the manual design process and it also limits the time needed by human experts.

Another approach using interactive genetic algorithms and a machine learning tool called *Dynamic Optimization* has been developed by Doijode et al. (2022). It uses a Naive-Bayes classifier to evaluate the design based on automatically generated CFD results. The classifier minimizes cavitation and maximizes the efficiency by finding the most appropriate compromise. The developed methods allow a

reduction of calculations by 30 %, but they are also not fully eliminated.

In order to completely avoid computational efforts in the design phase, Li et al. (2023) and Shora et al. (2018) used artificial neural networks (ANNs) to find the thrust and moment coefficients of a new optimum propeller design. While Li et al. (2023) compared different machine learning approaches next to ANNs, Shora et al. (2018) considered cavitation data and focused on a comparison of different structures of ANNs. They could show that in their network architecture there is no difference between using one ANN for each output variable versus using one ANN for all output variables. In both papers a limited amount of training data is used and a grid study was not conducted, which is going to be in focus of the research study presented.

## 3 DEVELOPMENT OF THE DATASET

In order to generate a comprehensive high quality data set RANS calculations are performed. For this purpose a parameterized geometry of a propeller is modified with the propeller tool in the CAE program CAESES developed by Friendship Systems AG to obtain variate designs. The radial distribution of the chord, camber, pitch, skew and number of blades is automatically altered and a CAD file of each propeller geometry with shaft and hub is created. The ranges for the parameters are shown in Table 1. The RANS solver STARCCM+ 17.02 is used for the computation of the hydrodynamic properties of the developed geometries. The computations are performed for the open water test condition. Therefore, only one blade is considered in the simulation and the periodic symmetry condition is applied. This setup allows a significant reduction of the computational effort. A stationary calculation with a moving reference frame is used. This way only the flow in the rotating region is rotated which also saves computational effort.

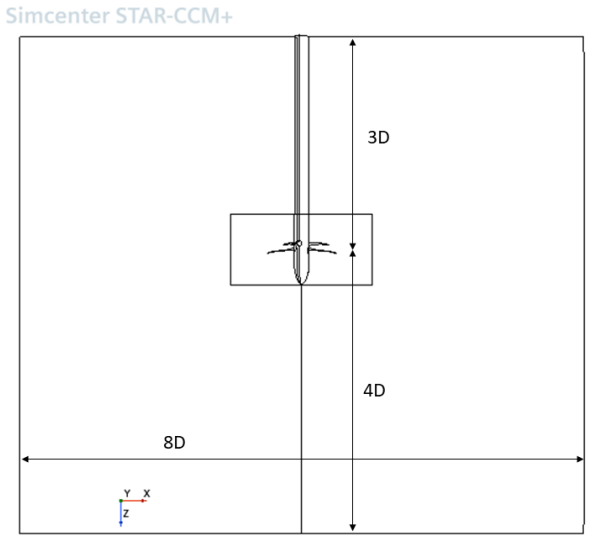


**Figure 1: Dimensions of calculation area in StarCCM+**

In Figure 1 the set up for the calculation is shown. The length of the computation domain is three times the propeller diameter in front of the propeller and four times behind it. The total radius of the computation domain is four times the propeller diameter (Tu 2019). The inflow velocity is set according to the respective advance ratio $J$ and at the outlet the pressure is set to zero.

**Table 1: Ranges for altered geometry parameters**

| Parameter | Min | Max |
|---|---|---|
| Chord | 0.25 | 0.4 |
| Camber | 0.01 | 0.06 |
| Pitch | 1.35 | 1.75 |
| Skew | 0.1 | 0.35 |
| Number of blades | 3 | 6 |

### 3.1 Mesh topology

The mesh is automatically adapted according to the propeller diameter. However, in the data set used the diameter is kept constant. To obtain a gap-free surface of the propeller a surface wrapper with part size $\frac{1}{100}D$ is used. The mesh is polyhedral with a base size $B = B(D)$ depending on the diameter and refined at the propeller surface with $\frac{2}{100}B$. The propeller leading and trailing edges are refined with $\frac{1}{250}B$ and a cylindrical volume of 2D is additionally refined with $\frac{1}{25}B$. All mesh variables are scalable with the base size which is set as $B = D$, $\frac{D}{2}$ or $\frac{D}{3}$.

A prism layer mesh is introduced in the boundary layer region. The wall function is applied and therefore the target $y^+$ is set to 30, as the calculations are carried out in full-scale size and with respective Reynolds numbers. The velocity $v_a$ is defined by the advance ratio $J$ and the stretch factor $s$ is set to 1.5. The chord length $c_L$ is approximated with $\frac{D}{2}$, $\mu$ is set to 0.00089 and $\nu$ is calculated with $\mu$ and $\rho$.

The following calculations are on the basis of the book about boundary layer theory by Schlichting (1987). With Equation 1 the reference velocity is calculated and then used to approximate the Reynolds number with Equation 2. The total prism layer thickness can then be estimated with Equation 3 and the thickness of the first prism layer is calculated with Equation 4. From there the necessary number of prism layers in the boundary layer is determined with Equation 5 and the thickness of the layer next to the wall with Equation 6. Then all necessary parameters for the prism layers are calculated. An offset volume of the propeller is refined depending on the last prism layer thickness $t_n$ to obtain cells with similar sizes.

$$v_{ref} = \sqrt{v_a^2 + (0.7\frac{\pi}{D}n)^2} \tag{1}$$

$$Re = \frac{c_L v_{ref}}{\nu} \tag{2}$$

$$t = \frac{0.16}{R_e^{0.142857} c_L} \tag{3}$$

$$t_1 = \frac{y_+ \mu}{\rho\sqrt{\frac{v_{ref}^2}{2}} 0.027 Re^{-0.142857}} \tag{4}$$

$$n_L = \frac{\log \frac{t}{t_1}}{\log s} \tag{5}$$

$$t_n = s^{n_L} - t_1 s^{n_L - 1} \tag{6}$$

### 3.2 Grid study mesh

Within the mesh convergence grid study, the KCS propeller geometry is used as a test case. The calculated results are compared to those of the model open water test conducted by Sanada et al. (2018). The torque values at $J = 0.6$ and the open water diagram are compared. As the measurements are carried out in model scale the wall $y_+$ is adjusted to 1 or less. This way the ITTC guideline regarding the near-wall region is followed (ITTC 7.5-03-02-03). The results of thrust and torque values with the corresponding errors at $J = 0.6$ are shown in Table 2. Three different mesh base sizes relative to the propeller diameter are compared.

**Table 2: Torque and thrust values for different mesh base sizes at $J = 0.6$**

| Value | Target | $D$ | $\frac{D}{2}$ | $\frac{D}{3}$ |
|---|---|---|---|---|
| Thrust [N] | 6.268 | 6.561 | 6.553 | 6.538 |
| Error [%] | - | 4.7 | 4.5 | 4.3 |
| Torque [Nm] | 0.105 | 0.111 | 0.110 | 0.110 |
| Error [%] | - | 5.7 | 4.8 | 4.8 |
| Number of mesh cells in Millions | - | 0.2 | 0.8 | 2.1 |

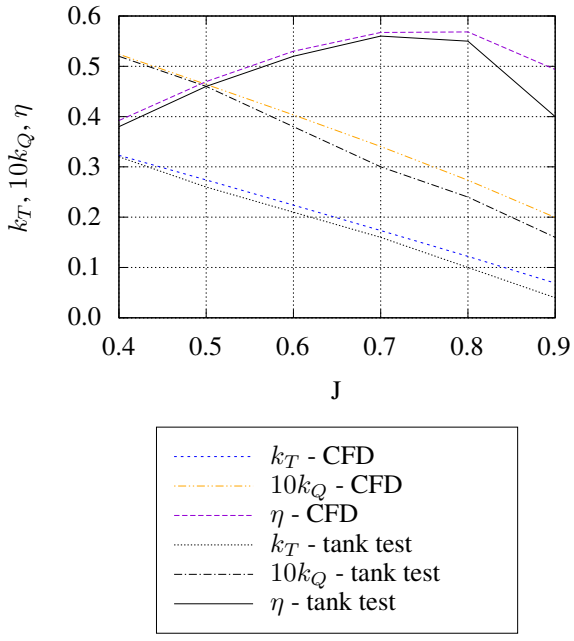


**Figure 2: Open water diagram for model tank test results and CFD calculation**

From $D$ to $\frac{D}{2}$ the cell number increases by factor 4 whereas the error is decreasing by 0.2 % and 0.6 % respectively. The error from $\frac{D}{2}$ to $\frac{D}{3}$ for one value decreases 0.2 % and for the other value by 0.0 % whereas the cell number increases almost by factor 3. As a result of the high cell number, the computation time rises significantly whereas the accuracy is almost the same. Thus, a mesh base size of $\frac{D}{2}$ is chosen for the calculations as a compromise and the quality of results should be achieved.

Figure 2 shows a comparison between the open water diagram measured at the model tank and the calculated one using a mesh base size of $\frac{D}{2}$. The average error for all points up to the maximum at $J = 0.8$ is 5.4 %. The $k_T$ values at $J = 0.9$ are near zero so that the small deviation would have a greater effect on the efficiency, and that area is usually not considered in propeller design.

### 3.3 Output data and treatment

The setup has been described and 500 CAD-geometries for each blade number have been used in order to obtain the open water diagrams and pressure distributions over the blade. Automated post-processing of this data needs to be performed in order to identify valid points for training the networks. As a first step only open water diagrams are used where the efficiency is above zero and below one. As the second step the remaining points are processed with a k-means clustering algorithm to check for outliers. With that algorithm, k point clouds - so called clusters - around k centroids are formed from the data. For this dataset of the open water diagrams an algorithm with k = 2 centroids has the best silhouette score and is thus used for the clustering. An example of the two colored clusters is shown in Figure 3.

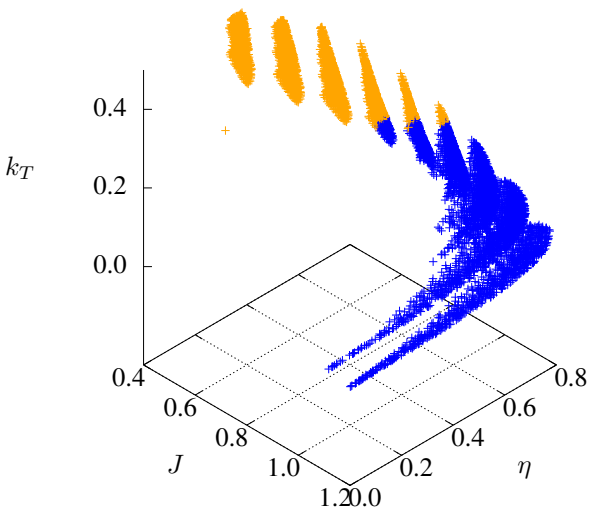


**Figure 3: Picture of two clusters in k-means-clustering**

The 50 farthest away points from those cluster mid points have been examined manually and no invalid data could be found this way. The quality of the training data is sufficient. Multiple datasets have been created and in this work a small dataset with 300 points, a medium dataset with 3.330 points and a large dataset with 10.030 points are used. A scatter plot of all open water diagrams for the medium dataset with 3.330 points is shown in Figure 4.

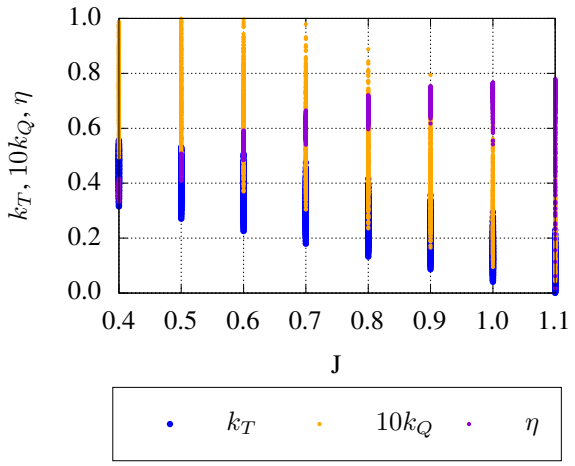


**Figure 4: Scatter plot of medium openwater curve data**

The pressure distribution is used to obtain an estimated cavitation area. As an approximation the ratio between points below vapour pressure and all points is calculated. An example of a pressure distribution is shown in Figure 5 and the according wall shear stress distribution in Figure 6.

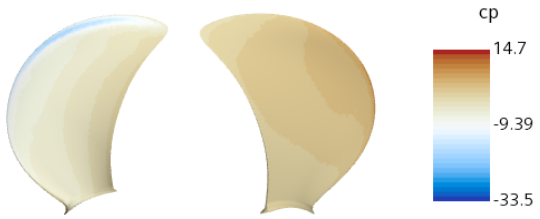


**Figure 5: Example of a pressure distribution on the face (left) and back (right) of a blade**

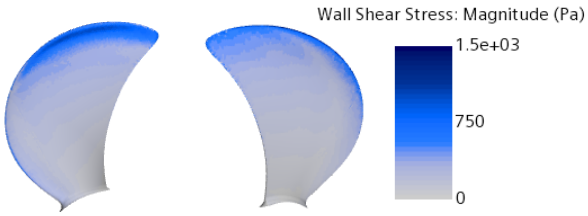


**Figure 6: Example of the wall shear stress on the face (left) and back (right) of a blade**

## 4 NEURAL NETWORK SETUP

For the neural networks simple feedforward neural networks are used. They consist of a certain amount of layers with a certain amount of neurons in them, beginning with an input layer where the data is fed in, then one or several hidden layers and an output layer where processed data is returned. The goal of the network is to adjust its weights with every training epoch in a way that a defined loss is minimized. A commonly used loss is the mean squared error loss (MSE), which is used here as well. The calculation is shown in Equation 7.

$$MSE(f(X), \hat{f}(X)) = \frac{1}{N}\sum_{n=1}^{N}(f(X_n) - \hat{f}(X_n))^2 \quad (7)$$

In order to find the operating point and then the associated geometry a two step neural network is used. The first network uses the number of blades, the torque or thrust coefficient $k_Q$ or $k_T$ and the respective $k_T^*$ or $k_Q^*$, see Equations 8 to 11, depending on whether the design is done via the ship resistance $R_T$ or the installed power $P_D$ and the relative rotative efficiency $\eta_R$.

$$k_T = \frac{R_T}{\rho n^2 D^4 (1-t)} \quad (8)$$

$$\frac{k_T}{J^2} = \frac{R_T}{\rho D^2 (1-t) v_s^2 (1-w)^2} = k_T^* \quad (9)$$

$$k_Q = \frac{\eta_R \frac{P_D}{2\pi n}}{\rho n^2 D^5} = \frac{\eta_R P_D}{2\pi \rho n^3 D^5} \quad (10)$$

$$\frac{k_Q}{J^3} = \frac{\eta_R P_D}{2\pi \rho D^2 v_s^3 (1-w)^3} = k_Q^* \quad (11)$$

As calculation variables, the necessary terms for estimating the load curve are used depending on whether the design is determined using the ship resistance and corresponding thrust or installed power and corresponding torque. The diameter, number of blades, design speed and wake fraction are needed for the calculation in both cases. Additionally, a maximum cavitation area and an aspired motor speed are given. The differing calculation variables are resistance and thrust deduction factor for the thrust as well as the installed

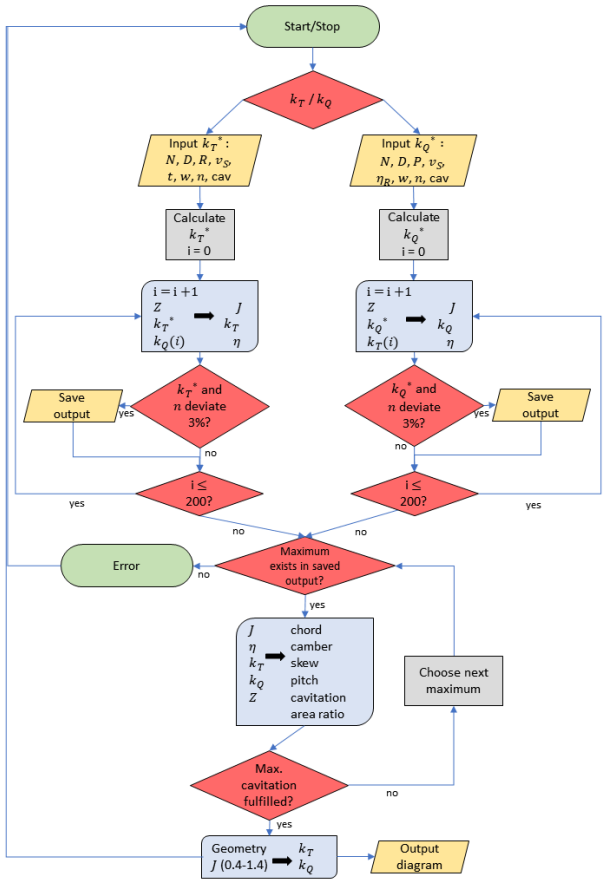


**Figure 7: Flow diagram of the output architecture with neural network blocks shown as rectangles with two rounded corners**

power and relative rotative efficiency for the power estimation. From this the input variables for the neural network $k_T^*$ and $k_Q^*$ are calculated and fed in together with the number of blades and $k_Q$ or $k_T$, respectively. The output variables are the respective thrust or moment coefficient and the advance ratio for the operating point.

In the second neural network the operating point is used as input variable together with the other input variables such as number of blades. The output is a parameterized geometry description of the blade and the expected cavitation area at the operating point. From the geometry an open water diagram is created. A flow diagram of the output algorithm with network architectures is shown in Figure 7.

### 4.1 Tuning the network parameters

The tuning of the network parameters is conducted using the python library Ray Tune. In general, there are several parameters to be tuned: the number of hidden layers, the number of neurons within them, the activation function and optimizer used as well as a dropout probability, which is used to prevent neural networks from overfitting. This is especially important for the training of networks with little amount of data. All tunable parameters and their ranges can be found in Table 3.

**Table 3: Parameters for the optimization of neural networks**

| Parameter | Variations |
|---|---|
| Number of hidden layers | 2 |
| Neurons per layer | 8, 16, 32 |
| Activation function | ReLU, LeakyReLU, Softplus, Tanh |
| Optimizer | SGD, Adam, rmsprop |
| Learning rate | 0.00001 to 0.1 |
| Batch size | 1, 2, 4, 8 |
| Dropout probability | 0, 0.1, 0.5 |

For tuning of the network random parameter combinations are trained with the data for 50 epochs. After training those epochs, the resulting loss is compared and the network with the smallest loss is chosen for further training. The best parameter fit for the corresponding dataset-sizes is shown in Table 4. It can be seen that most networks perform best with the activation function ReLU and the optimizer SGD. They also have the same number of hidden layers and number of neurons within the layers. This is important to note as the parameters could be used as a starting point for tuning networks with physically similar problems.

### 4.2 Test with unknown data

For checking the performance of the neural networks regarding unknown data, several data points have been excluded in all of the train- and test datasets. They were used as input variables to the trained networks and the output has been compared to the actual values. In general the errors of those output variables are getting smaller with increasing the amount of training data. The MSE for different variables are shown in Tables 5 to 7.

**Table 5: MSE on test data for different training datasets using $k_T^*$**

| Parameter | small | medium | large |
|---|---|---|---|
| $J$ | 0.0188 | 0.0192 | 0.0133 |
| $\eta$ | 0.0323 | 0.0299 | 0.0143 |
| $k_T$ | 0.0342 | 0.0287 | 0.0141 |
| Total | 0.0284 | 0.0259 | 0.0139 |

**Table 6: MSE on test data for different training datasets using $k_Q^*$**

| Parameter | small | medium | large |
|---|---|---|---|
| $J$ | 0.0363 | 0.0316 | 0.0135 |
| $\eta$ | 0.0416 | 0.0210 | 0.0105 |
| $k_Q$ | 0.0985 | 0.0551 | 0.0390 |
| Total | 0.0588 | 0.0359 | 0.0210 |

**Table 4: Tuned neural network parameters**

| NN | Hidden layers | Neurons | Activation Function | Optimizer | Learning rate | Batch size | Dropout probability | MSE train | MSE test |
|---|---|---|---|---|---|---|---|---|---|
| $k_T$ - small | 2 | 16, 32 | ReLU | RMSprop | 0.060126 | 1 | 0 | 0.0007 | 0.0021 |
| $k_T$ - medium | 2 | 16, 32 | ReLU | SGD | 0.099461 | 1 | 0 | 0.0007 | 0.0006 |
| $k_T$ - large | 2 | 16, 32 | ReLU | SGD | 0.099461 | 1 | 0 | 0.0004 | 0.0004 |
| $k_Q$ - small | 2 | 16, 32 | ReLU | SGD | 0.099461 | 1 | 0 | 0.0017 | 0.0022 |
| $k_Q$ - medium | 2 | 16, 32 | ReLU | SGD | 0.060126 | 1 | 0 | 0.0017 | 0.0019 |
| $k_Q$ - large | 2 | 16, 32 | ReLU | SGD | 0.099461 | 1 | 0 | 0.0010 | 0.0009 |
| $geometry$ - small | 2 | 8, 32 | LeakyReLU | RMSprop | 0.062980 | 2 | 0 | 0.0017 | 0.0017 |
| $geometry$ - medium | 2 | 16, 32 | ReLU | SGD | 0.099461 | 1 | 0 | 0.0019 | 0.0019 |
| $geometry$ - large | 2 | 16, 32 | ReLU | SGD | 0.099461 | 1 | 0 | 0.0019 | 0.0019 |

**Table 7: MSE on test data for different training datasets for the geometry**

| Parameter | small | medium | large |
|---|---|---|---|
| Cavitation Area | 1.4688 | 0.6157 | 0.5476 |
| Total | 0.3288 | 0.1431 | 0.1361 |

Figure 8 shows the corresponding results for predicting the cavitation area when different training datasets are employed. The line would be the ideal output to the input data. The results clearly show that the higher the amount of training data, the more accurate the output. This is especially important for creating neural networks capable of computing the whole pressure distribution over the blade, as the cavitation area is only a simplified part of it.

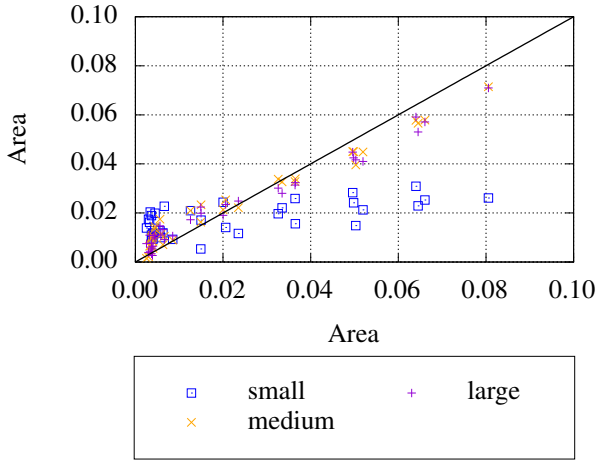


**Figure 8: Cavitation area - comparison of training dataset sizes**

## 5 COMPARISON TO EXAMPLE VESSEL

Depending on the available data for the torque or thrust load curve according to Equation 8 and 11, the trained networks are used to develop a parameterized geometry, calculate the open water curve and estimate the operating point. At the output a deviation of 3 % of the input $k_T^*$ or $k_Q^*$ and of the aspired motor speed are allowed. If a thrust load curve with $k_T^*$ is used, the input of $k_Q$ is varied and the result with the highest efficiency considered the solution. This is done to maximize the efficiency. The same applies to a variation of $k_T$ if a torque load curve with $k_Q^*$ is used. The operating point then is the point with maximum efficiency while only deviating 3 % of the input. The input data for the networks is shown in Table 8. The maximum allowed cavitation area of 2.5 % is chosen according to the Burill cavitation diagram with the suggested limit for merchant ship propellers (Burill et al. 1963).

**Table 8: Example ship data to find the operating point for "Vessel"**

| Parameter | "Vessel" |
|---|---|
| Number of blades | 4 |
| Diameter | 3 m |
| Maximum allowed cavitation area | 2.5 % |
| Velocity | 6.69 m s$^{-1}$ |
| Wake fraction | 0.25 |
| Resistance | 93.47 kN |
| Thrust deduction fraction | 0.1 |
| Shaft power | 720 kW |
| Rotational speed | 130 |
| Relative rotative efficiency | 1 |

From this data the respective $k_T^*$ and $k_Q^*$ values are calculated and shown as load curves in Figures 9 to 11. From the output parameters of the neural network the geometry is generated as a CAD file in CAESES and then the open water diagrams are calculated with STARCCM+. The open water test diagrams deviate slightly from one another as the computations are conducted for different geometries. In general all training dataset sizes show good results. The shown efficiency has been calculated from $k_T$ and $k_Q$, otherwise the efficiency has large errors for the small dataset. For the large dataset the output efficiency of the neural network is even slightly better than the calculated efficiency leading to an overall open water diagram MSE of 0.0001. This shows that enough training data has to be available for complex physical problems.

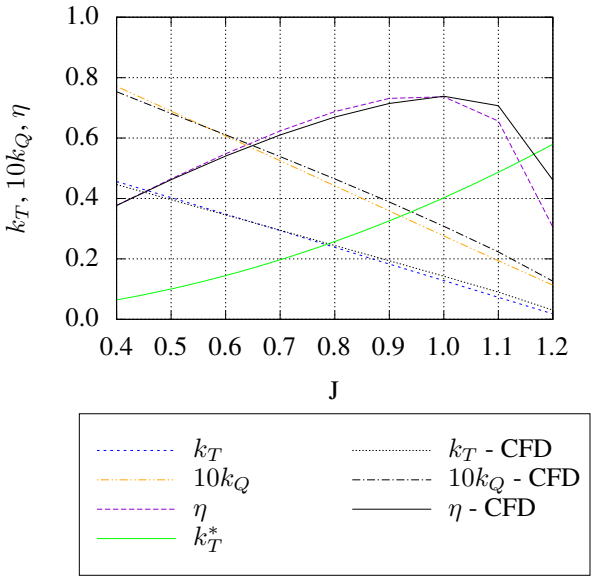


**Figure 9: Open water diagram small dataset**

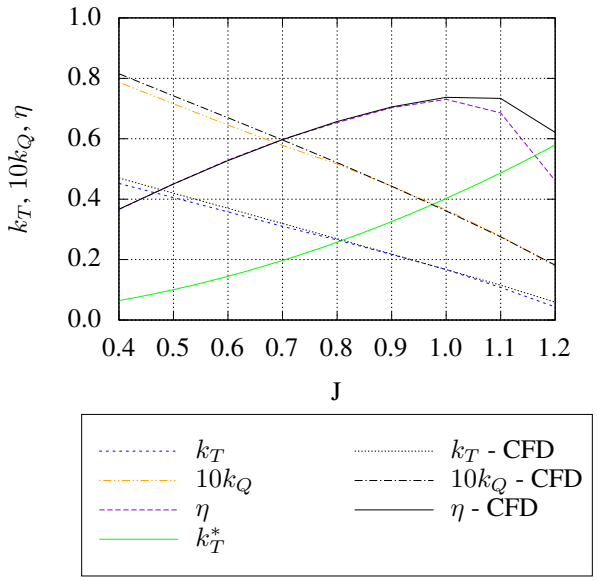


**Figure 10: Open water diagram medium dataset**

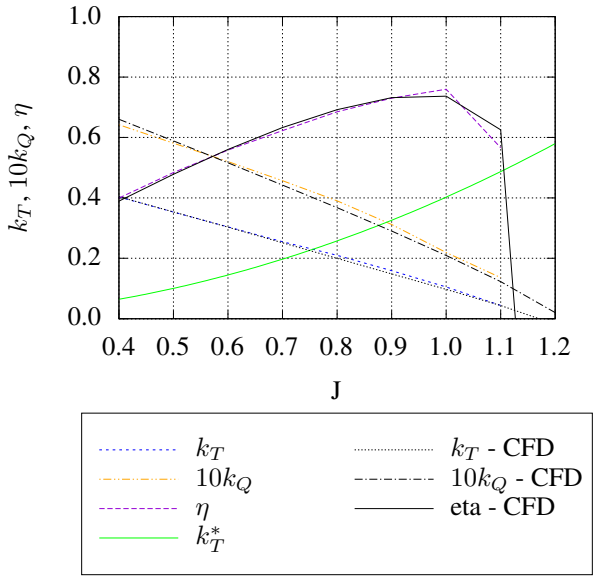


**Figure 11: Open water diagram large dataset**

**Table 9: MSE of the open water diagram of the example vessel for different training datasets**

| Parameter | small | medium | large |
|---|---|---|---|
| Total with $k_Q$ | 0.0015 | 0.0008 | 0.0009 |
| - up to max $\eta$ | 0.0002 | 0.0001 | 0.0001 |
| Total with $k_T$ | 0.0012 | 0.0012 | 0.0003 |
| - up to max $\eta$ | 0.0002 | 0.0002 | 0.0001 |

Table 9 includes the deviations between the calculated open water diagrams using NN and the predicted using CFD, for the whole open water diagram as well as for the open water diagram until the maximum efficiency. In Table 10 the operating points resulting from the load curves for the networks and the CFD-simulation outputs are compared. The estimations of the operating point of the open water diagrams shown in Figures 9 to 11 are based on the calculation using the ship resistance ($k_T^*$). Using the installed power leads to similar results.

**Table 10: Operating point of the example vessel from neural network and CFD data calculated with $k_T^*$**

| Parameter | small | | medium | | large | |
|---|---|---|---|---|---|---|
| | NN | CFD | NN | CFD | NN | CFD |
| $J$ | 0.77 | 0.79 | 0.79 | 0.81 | 0.75 | 0.75 |
| $k_T$ | 0.28 | 0.25 | 0.28 | 0.26 | 0.24 | 0.22 |
| $10k_Q$ | 0.55 | 0.47 | 0.55 | 0.51 | 0.44 | 0.40 |
| $\eta$ | 0.62 | 0.66 | 0.66 | 0.66 | 0.66 | 0.66 |

With Table 9 it can be shown that the error is decreasing with an increasing amount of training data. According to the results of the present study, at least 3.330 data points are required to obtain physically plausible results. They can even be improved by extending the dataset to 10.030 points. In Table 11 the output cavitation areas of the NNs and the CFD calculations are shown. All results of the NNs deviate distinctly which might lead to the usage of even larger datasets. A more plausible result is obtained with larger datasets, but there is also no clear alignment with the CFD results.

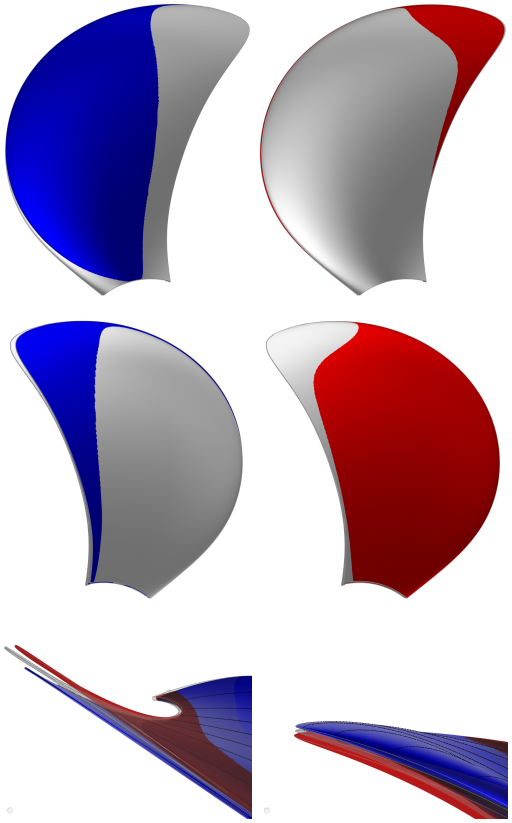

**Figure 12: Axial view of pressure and suction side of the geometries of the designed propellers from small (grey), medium (blue) and large data set (red).**

**Table 11: Cavitation area of the example vessel from neural network and CFD data**

| Parameter | small | | medium | | large | |
|---|---|---|---|---|---|---|
| | NN | CFD | NN | CFD | NN | CFD |
| Cavitation area ratio [%] | 2.3 | 0.1 | 1.4 | 0.2 | 0.6 | 0.1 |

The geometries for the three training data set sizes only vary slightly. There is a difference in $J$ from $0.75$ to $0.81$, which slightly shows in the according pitch (a pitch of 1.4 versus 1.5). All other geometry parameters are similar. A comparison of the blades is shown in Figure 12.

## 6 CONCLUSION

In the present work it is shown that a high amount of training data improves the neural networks performance for predicting the operating point, developing the parameterized geometry and estimating the cavitation area. Based on the results of the study, at least an amount of 3.330 points is necessary to obtain plausible output values. A slightly smaller error for some of the output values is obtained using 10.030 data points. This is also important in the case that the full pressure distribution shall be predicted by the networks. It might be necessary to use even more data because the problem is turning out to be more complex. For problems of similar complexity, for example other parameterized propellers with associated CFD-simulation data, the same approach could be used and should lead to comparable results. It could even be possible to use the best network parameters from this dataset as a starting point for training other physically related networks with a similar structure.

## REFERENCES


Burill, L.C. & Emerson, A. (1963). 'Propeller cavitation: Further tests on 16in. propeller models in the King's College cavitation tunnel'. International Shipbuilding Progress, **10**(104), pp. 119–131.

Doijode, P.S., Hickel, S., van Terwisga, T. & Visser, K. (2022). 'A machine learning approach for propeller design and optimization: Part II'. Applied Ocean Research **124**, 103174.

Gypa, I., Jansson, M., Wolff, K. & Bensow, R. (2021). 'Propeller optimization by interactive genetic algorithms and machine learning'. Ship Technology Research (Schiffstechnik) **70**(1), pp. 56–71.

Gypa, I., Jansson, M. & Bensow, R. (2023). 'Marine propeller optimisation through user interaction and machine learning for advanced blade design scenarios'. Ships and Offshore Structures **0**, pp. 1–17.

Li, L., Chen, Y., Qiang, Y., Zhou, B. & Chen, W. (2023). 'Construction and application of numerical diagram for high-skew propeller based on machine learning'. Ocean Engineering **278**, 114480.

Sanada, Y., Kim, D.-H., Sadat-Hosseini, H., Toda, Y., Simonsen, C. & Stern, F. (2018). 'Experiment and Numerical Simulation for KCS Added Powering in Regular Head/Oblique Waves'. Proceedings of the 32nd Symposium on Naval Hydrodynamics. Hamburg, Germany.

Shora, M.M., Ghassemi, H. & Nowruzi, H. (2018). 'Using computational fluid dynamic and artificial neural networks to predict the performance and cavitation volume of a propeller under different geometrical and physical characteristics'. Journal of Marine Engineering & Technology **17**(2), pp. 59–84.

Schlichting, H. (1987). Boundary-Layer Theory. 9th ed. Springer Berlin Heidelberg.

Tu, T.N. (2019). Numerical simulation of propeller open water characteristics using RANSE method. Alexandria Engineering Journal **58**(2), pp. 531–537.

Vardhan, H., Volgyesi, P. & Sztipanovits, J. (2021). Machine learning assisted propeller design. Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems. Nashville, Tennessee, USA.