EUROGEN 2023
15th ECCOMAS Thematic Conference on
Evolutionary and Deterministic Methods for Design, Optimization and Control
N. Gauger, K. Giannakoglou, M. Papadrakakis, J. Periaux (eds.)
Chania, Crete Greece, 1–3 June 2023

DNN-ASSISTED EA-BASED SHAPE OPTIMIZATION OF A HYDRAULIC TURBINE

Marina G. Kontou, Varvara G. Asouti, Xenofon S. Trompoukis and Kyriakos C. Giannakoglou

Parallel CFD & Optimization Unit (PCOpt), School of Mechanical Engineering,
National Technical University of Athens (NTUA), Athens, Greece
e-mails: {mkontou,vasouti}@mail.ntua.gr, xeftro@gmail.com, kgianna@mail.ntua.gr

Keywords: CFD, Shape Design/Optimization, Deep Neural Networks, Evolutionary Search, Surrogates

Abstract. *This paper proposes the use of DNNs as a means to reduce the number of domains that need to be simulated in multi-row CFD runs in the context of a shape optimization based on stochastic, population-based algorithms. This is demonstrated in the shape optimization of a hydraulic turbine that consists of stay vanes, guide vanes, runner blades and the draft tube. Only the runner blades, the shape of which are parameterized by a volumetric NURBS lattice, are allowed to vary during the optimization. For the CFD analysis, the NTUA's GPU-accelerated multi-row flow solver PUMA, which solves the RANS equations for incompressible flows, coupled with the Spalart-Allmaras turbulence model, is used. The CFD simulation of such a multi-row configuration makes use of the mixing plane technique for the interaction between adjacent rotating and stationary domains. Herein, a DNN-based surrogate to the mixing plane technique is used to avoid the CFD solution into the domains of the stay and guide vanes and the draft tube. The DNN is trained based on the coordinates of the (free) control points of the NURBS lattice and predicts the averaged quantities exchanged at the interfaces (mixing surfaces) between the guide vanes and the runner as well as the runner and the draft tube. The optimization is carried out by means of metamodel-assisted evolutionary algorithm (MAEA) enhanced by the Principal Components Analysis (the EASY platform developed by the group of authors). This MAEA has a dual role as it is used both for the definition of the optimal DNN architecture and the shape optimization of the runner.*

1 INTRODUCTION

During the last years, DNNs have been widely used in assisting CFD analysis and optimization problems due to their ability to detect input data properties and patterns and make low-cost predictions. Indicatively, in [1], the group of authors of this paper proposed a DNN-based surrogate for turbulence closure of the RANS equations. The DNN was used to replace the numerical solution of the turbulence and transition models by providing the turbulent viscosity field in each pseudo-time iteration of the RANS solver. The less demanding DNN-assisted RANS solver was demonstrated in the shape optimization of a transonic turbine blade and a car model. A physics-informed convolutional neural network was used for flow predictions in [2]; emphasis was laid on the preservation of the physical laws. In [3], a deep Long Short-Term Memory network was used to predict the turbine flow characteristics based on real machine operation data from a hydropower plant.

In this work, DNNs are used to reduce the number of simulation domains in the case of multi-row CFD simulations. In specific, a DNN-based surrogate replaces the Rotor-Stator Interaction (RSI) technique, i.e. the need for simultaneously simulating two successive rows and the iterative exchange of data along the interfaces through the mixing plane technique. A number of multi-row CFD runs is used to collect data and train the DNNs on them, so that, during the optimization of the runner, only the latter undergoes a CFD analysis, whereas the other domains manifest their presence through the trained DNNs. To showcase the proposed methodology, a computationally demanding hydraulic turbine application is selected.

Nowadays, hydropower is considered the most important source of renewable energy. In order to be competitive and meet the tight delivery schedule of new products, suppliers rely on CFD tools and optimization methods for the design of hydraulic turbines and their components. At the same time, the increased demand for flexibility in the hydropower plants pushes towards an extended operating range of hydraulic turbines. When operating at off-design loads, the flow through the turbine may become unstable and presents instabilities (such as pressure pulsations, structural vibrations etc.) which compromise the smooth and safe operation of the machine, [4].

Herein, the CFD-based shape optimization of a hydraulic turbine (consisting of stay, guide vanes, runner blades and the draft tube) aims at reducing the amplitude of pressure pulsations between the runner and the guide vanes while also ensuring that no cavitation occurs in the runner. An evolutionary algorithm assisted by surrogate evaluation models (Metamodel-Assisted Evolutionary Algorithm; MAEA) and the Principal Component Analysis (PCA), [5], is used to optimize the runner blade shape. The evaluation of each candidate solution is carried out using the in-house GPU-accelerated solver PUMA, [6], which solves the RANS equations for incompressible flows, coupled with the Spalart-Allmaras turbulence model. The mixing plane technique is used to model guide vanes-runner and runner-draft tube interactions when running the CFD code through the entire turbine (this code will be referred to as the CFD^{F}). The idea this paper is based upon is to train and use DNN-based surrogates to the mixing plane technique, before and after the only domain to be solved by the CFD tool (that of the runner), to overcome the simultaneous CFD solution into the domains of stay and guide vanes as well as the draft tube; the new model will be abbreviated to CFD^{R} as it simulates the flow by the CFD code only in the runner (=R) domain; the two DNNs are indispensable parts of CFD^{R}). "Optimal" hyper-parameters of the DNN must be found and this is also based on the above-mentioned MAEA, which thus undertakes two different roles in this work.

2 BACKGROUND METHODS AND TOOLS

2.1 Optimization tool (MAEA)

The shape optimization of the runner geometry as well as the optimization of the DNN hyperparameters considered in this work are carried out by means of the MAEA, enhanced by the PCA. This is implemented in the Evolutionary Algorithms SYstem (EASY) optimization platform, [7], developed by the PCOpt/NTUA. During each generation, the (μ, λ) EA, maintains and updates three populations, namely that of λ offspring, that of μ parents and the set of the (at most e) best so-far solutions. EASY uses on-line trained surrogate evaluation models or metamodels that replicate the problem specific model (herein, the CFD tool). The first few generations run as in a standard EA (used to collect the first T^{MM} evaluated individuals in the dynamically expanded database or DB_{EA}). Within each subsequent generation, a single personalized metamodel is trained for each offspring, on its neighboring already evaluated (during the evolution) individuals and used to pre-evaluate it at minimal cost. After pre-evaluating all the population members on the metamodels, only a few promising (λ_e) individuals in each generation are re–evaluated on the CFD tool. Radial Basis Functions (RBF) networks are used as metamodels.

Engineering optimization problems with many design variables, such as the one tackled herein, usually suffer from the "curse of dimensionality". To alleviate this problem, the Kernel PCA is additionally employed in each generation. The PCA of the offspring population is used to control the evolution operators and/or prune the number of the metamodels' input units. With regard to the former, the variances resulted from the PCA are used to transform the parents into a new feature space. Crossover and mutation take place in the feature space and the new offspring are transformed back into the design space. Regarding the use of the PCA to improve the prediction accuracy of metamodels, the training patterns, used for the personalized metamodel of each population member are transformed to the feature space whereas some of the transformed inputs, those along directions with the smaller variances, are truncated. By doing so, the metamodels yield improved predictions and the overall algorithm converges faster.

2.2 CFD analysis tool

The CFD analysis tool used in this work is the in-house GPU-accelerated s/w PUMA, [6, 8], which solves the RANS equations for incompressible fluids using the artificial compressibility method. The steady residuals of the flow equations read

$$R_{U_n} = \frac{\partial f^{\text{inv}}_{nk}}{\partial x_k} - \frac{\partial f^{\text{vis}}_{nk}}{\partial x_k} + \mathcal{S}_n = 0 \tag{1}$$

where $U_n = [p,\ v_1,\ v_2,\ v_3]^T$ is the flow variables array with p the kinematic pressure and $v_k\ (k=1,2,3)$ the absolute Cartesian velocity components. The inviscid ($\mathbf{f}^{\text{inv}}_{\mathbf{k}}$), viscous ($\mathbf{f}^{\text{vis}}_{\mathbf{k}}$) fluxes and the source terms ($\mathcal{S}$) are written as

$$\mathbf{f}^{\text{inv}}_{\mathbf{k}} = \begin{bmatrix} \beta^2 w_k \\ w_k v_1 + p\delta_{1k} \\ w_k v_2 + p\delta_{2k} \\ w_k v_3 + p\delta_{3k} \end{bmatrix}, \quad \mathbf{f}^{\text{vis}}_{\mathbf{k}} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \end{bmatrix}, \quad \mathcal{S} = \begin{bmatrix} 0 \\ \varepsilon_{1\ell k}\omega_\ell v_k \\ \varepsilon_{2\ell k}\omega_\ell v_k \\ \varepsilon_{3\ell k}\omega_\ell v_k \end{bmatrix} \tag{2}$$

where β is a parameter corresponding to a (constant) artificial speed of sound, w_k the relative velocity components, ω the rotational velocity vector and δ_{km}, $\varepsilon_{i\ell k}$ the Kronecker and permutation symbols. The stress tensor is $\tau_{km} = (\nu + \nu_t)\left(\frac{\partial v_k}{\partial x_m} + \frac{\partial v_m}{\partial x_k}\right)$ where ν and ν_t stand for the kinematic and turbulent kinematic viscosities, respectively. Eqs. 1 are loosely coupled with the Spalart-Allmaras turbulence model equation, [9].

The interaction between adjacent rotating and stationary domains (RSI) is modeled using the mixing plane technique. According to this, in the CFD^F tool, the spanwise distribution of circumferentially averaged (mixed-out) flow variables $\hat{V} = [\hat{p},\ \hat{v}_r,\ \hat{v}_\theta,\ \hat{v}_a]^T$ is communicated between the adjacent domains; v_r, v_θ and v_a stand for the radial, peripheral and axial components of the absolute velocity vector and the hat ($\hat{}$) symbol indicates circumferentially averaged quantities. Then, using $\hat{V}$ as well as the flow variables of each domain, the numerical flux to be imposed at each RSI mesh node is computed. Upon convergence of the flow equations, the numerical fluxes crossing the interface are conserved.

PUMA implements a vertex–centered finite volume approach on hybrid meshes consisting of tetrahedra, pyramids, prisms and/or hexahedra. A multi-stage Runge–Kutta scheme with implicit residual smoothing is used. The inviscid fluxes are discretized using a second-order Roe's upwind scheme. The software runs on GPUs with minimal memory requirements and increased parallel efficiency. This is attributed to the Mixed Precision Arithmetic technique according to which, the quantities of the left-hand-side are computed in double, though stored in single precision.

2.3 Shape parameterization and mesh deformation tool

The runner blade shape is controlled using an in-house free-form deformation technique based on volumetric NURBS, [10]. The same tool also controls the volume mesh deformation, avoiding the use of a mesh displacement tool. This tool was made for turbomachinery applications, so it takes periodicity into account and, when updating the blade shape, ensures hub and shroud axisymmetry. The latter is achieved through an intermediate coordinate system transformation, [10]. The control lattice is primarily defined in the new coordinate system and, then, transformed into the Cartesian one.

The $11 \times 3 \times 5$ NURBS control lattice used to parameterize the runner is shown in Fig. 1. The coordinates of some of these control points are selected as the design variables. In specific, 9 out of the 11 series of control points in the streamwise direction are allowed to vary in both the streamwise and pitchwise directions giving rise to 180 design variables in total.

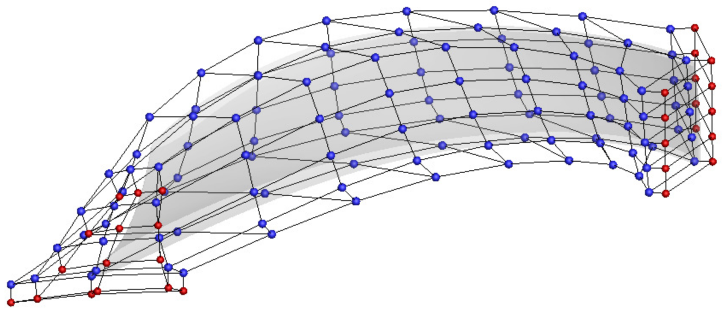

Figure 1: The runner blade enclosed into the NURBS control lattice. Blue points are allowed to vary during the optimization while red ones remain constant.

3 DNN-BASED SURROGATE FOR THE MIXING PLANE TECHNIQUE

The hydraulic turbine considered in this work consists of 21 stay, 21 guide vanes, 9 runner blades and a draft tube, Fig. 2. The optimization aims at minimizing the difference of the max. and min. pressure ($F = [P_{\max} - P_{\min}]_{Probe}$) at a point (probe) located between the guide vanes and the runner (at a specified radial and axial position). The temporal variation of this point can easily be seen as a spatial distribution along the corresponding circumference. Constraints that (a) retains the torque produced by the turbine and (b) overcomes cavitation on the runner blade are imposed.

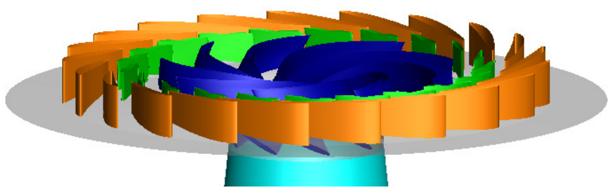

Figure 2: Perspective view of the baseline turbine geometry with the stay vanes (orange), guide vanes (green), runner blades (blue) and (just the very first part of) the draft tube (ice blue).

Given that all the aforementioned quantities of interest (objective and constraints) result from flow data computed in the runner domain, the idea is to simulate the flow only in the runner rather than in the entire turbine. Using the terminology introduced in section 1, the plan is to use the CFD^{R} (instead of the CFD^{F}), tool within the MAEA-based search. This will reduce the computational cost of each evaluation and, thus, the overall optimization turnaround time.

To do so, boundary conditions that replicate the presence of the stators (stay and guide vanes) and the draft tube must be imposed at the inlet and outlet of the runner domain. Since the RSI is modelled via the mixing plane technique (schematically illustrated in Fig. 3), two DNN–based surrogates of the RSI are used to predict the distributions of the flow quantities communicated between the adjacent domains at these interfaces. In specific, the two DNNs undertake the prediction of the axial distribution of the runner inlet flow variables (DNN$_1$) and the radial one of the runner exit flow variables (DNN$_2$), Fig. 4.

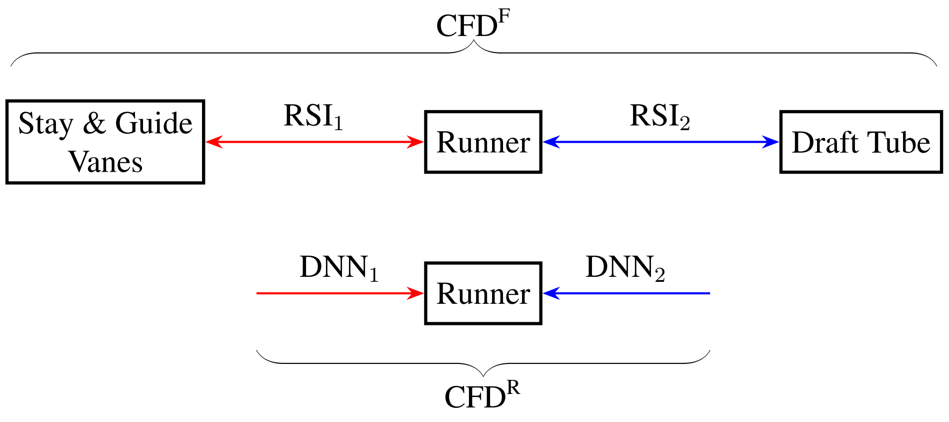

Figure 3: Schematic representation of the exchange of information (numerical fluxes) between adjacent rotating and stationary domains at the two interfaces (RSI) in the case of CFD^{F} (top), and CFD^{R} supported by the DNNs (bottom).

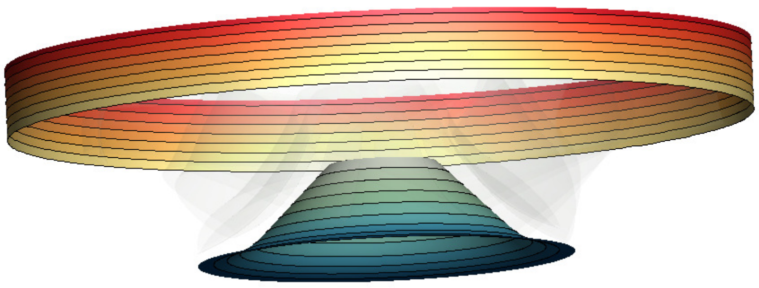

Figure 4: The radial inflow and axial outflow boundaries of the runner domain. The two interfaces of the runner with the guide vanes (top) and the draft tube (bottom) along with the zones used for averaging the flow quantities.

3.1 Configuring and training the two DNNs

The first step to be taken is to sample the runner's blade design space and generate the database ($\mathrm{DB_{DNN}}$) that the DNN will be trained on. $\mathrm{DB_{DNN}}$ should not be confused with the $\mathrm{DB_{EA}}$ supporting the MAEA. For each value set of design variables, corresponding to a different runner geometry, the full turbine (with all the aforementioned components) is simulated using the the PUMA software. The large number of design variables (180) makes the sampling procedure challenging. A small number of samples which though ensures that the so-generated $\mathrm{DB_{DNN}}$ is suitable for the DNN training would be ideal. The Latin hypercube sampling technique is used and three $\mathrm{DB_{DNN}}$ sizes consisting of 50, 75 and 100 samples are generated. These will be denoted as $\mathrm{DB^{50}_{DNN}}$, $\mathrm{DB^{75}_{DNN}}$ and $\mathrm{DB^{100}_{DNN}}$, respectively. Later on, the corresponding trained DNNs will be assessed in terms of cost of the optimization run. Assuming that an evaluation on the $\mathrm{CFD^F}$ corresponds to one (1) cost unit, forming the $\mathrm{DB_{DNN}}$ costs as many cost units as the $\mathrm{DB_{DNN}}$ size.

The DNN training uses the coordinates of the NURBS' lattice control points that are allowed to vary as inputs and predicts the distributions of the flow quantities (velocity vector and pressure) at the RSI, to be used as the inlet and outlet conditions for the runner domain.

In this work, Fully Connected Neural Networks (FCNN) are considered. The hyperparameters of each network result from a $(\mu, \lambda) = (10, 30)$ MAEA-based optimization aiming at minimizing the DNN prediction error, [1]. The unknowns are the number of layers, the number of neurons per layer (in powers of 2) and the activation function in each layer. For the latter, the algorithm has to select among the Rectified Linear Unit (ReLU), Gaussian Error Linear Units (GELU), hyperbolic tangent (tanh) and sigmoid functions. The DNN setup and training is carried out in the TensorFlow framework using Python. The Adam optimizer, [11], is used. The use of an optimization algorithm to define the DNN hyperparameters enhances the DNN reliability for the upcoming shape optimization. In our case, the overall cost of the optimization of the hyperparameters of both DNNs, including the training itself, sums up to no more than 2 cost units.

The optimal DNNs configurations are summarized in Table 1. In all cases, the optimization selected the GELU activation function for all the hidden layers and the sigmoid one for the output layer. Though the number of layers is not affected by the number of training patterns, this is not the case for the number of neurons per layer. Nevertheless, the total number of the trainable parameters of each DNN are comparable. For the $\mathrm{DNN_1}$ these are about $20M$, while for the $\mathrm{DNN_2}$ about $\sim 5M$ (see last column of Table 1). This means that the DNN hyperparameters could have been optimized only once (for any $\mathrm{DB_{DNN}}$ size), followed by an independent training for each DB_{DNN} size, since the number of trainable parameters is, more or less, the same for all optimal configurations. The use of three different DNN architectures was decided to ensure a fair comparison between the performances of the DNNs during the shape optimization of the hydraulic turbine in Sec. 4.

Position	DB_{DNN}	Layers	Neurons/Layer	Parameters
Inlet (DNN_1)	50	7	4096, 4096, 1024, 64, 128, 256, 1024	22M
	75	6	2048, 4096, 1024, 4096, 256, 512	19M
	100	7	128, 64, 4096, 4096, 256, 1024, 64	22M
Outlet (DNN_2)	50	4	2048, 1024, 256, 1024	3M
	75	3	1024, 1024, 1024	5M
	100	4	2048, 2048, 32, 256	4.5M

Table 1: Optimal configurations for DNN_1 (inlet) and DNN_2 (outlet).

3.2 Assessment of the trained DNNs

The proposed DNN-based surrogates to the mixing plane conditions are initially applied and assessed in the baseline geometry. The DNNs trained on the DB^{50}_{DNN} are selected and used to predict the velocities and the pressure distributions at the RSI zones of the turbine. These are compared with those resulted from a CFD^F simulation, Fig. 5. The agreement is absolutely satisfactory; small discrepancies exist in some of the zones of the axial velocity profile at which values are overestimated by the DNN. The flow simulation in the runner domain using either CFD^F or the DNN (predicted) distributions yield similar objective function values; a percentage error of about 1.1%, with the DNN overestimating the objective function value is observed. Overall, the DNNs are considered reliable for use together with the CFD^R, in the shape optimization.

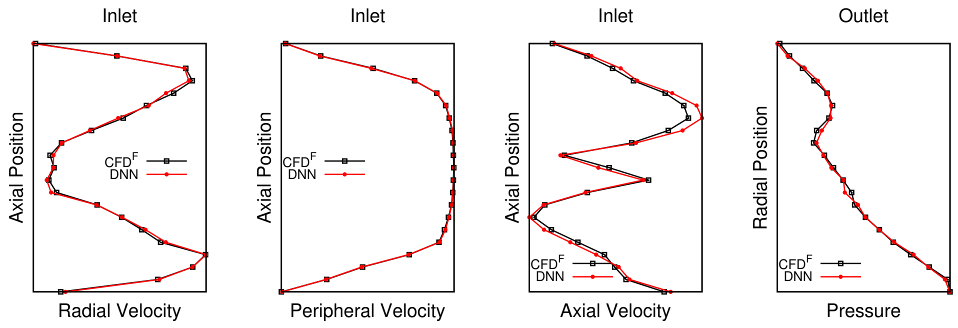

Figure 5: Comparison of the radial, peripheral, axial velocity and pressure (from left to right) between the CFD^F (black) and the DNN prediction (red).

4 SHAPE OPTIMIZATION STUDIES

This section summarizes the shape optimization studies using the proposed DNN-based surrogates for the mixing plane. For the sake of comparison, an additional run simulating the flow in the entire turbine domain (using CFD^F) is also presented. The target is to suppress pressure pulsations between the guide vanes and the runner by additionally retaining the torque produced by the turbine and ensuring that no cavitation occurs on the runner blade. The objective is quantified by the amplitude of the pressure field defined at a constant radius between the guide vane-runner interface and the runner leading edge. To avoid cavitation, the optimization should ensure that the min. pressure over the optimal runner blade exceeds that of the non-cavitated runner of the baseline geometry; this is a relative, rather than an absolute cavitation criterion.

In all studies, a $(\mu, \lambda) = (12, 20)$ MAEA is used. Its problem-agnostic metamodels (RBF networks) are activated after at least $T^{MM} = 50$ evaluations on the problem specific model (either CFD$^{\text{F}}$ or CFD$^{\text{R}}$), provided that at least 30 of them meet the constraints; these are all archived into the DB$_{\text{EA}}$. The previous criteria ensure that there will be enough data to train dependable metamodels. In generations that are assisted by metamodels, the best (according to objective function predictions on a properly trained metamodel) $\lambda_e \in [2, 4]$ population members are re-evaluated on the PSM; the selection of the λ_e value is related to the degree constraints are violated and the accuracy of the metamodels' prediction. The PCA is activated after the 3rd generation and the metamodels are trained using the 45 first principal components, pinpointed by the PCA of the current offspring population.

The computational cost of an evaluation based on CFD$^{\text{F}}$ (the entire turbine) is ~ 15 min. on a single A100 NVIDIA GPU (one cost unit). The CFD$^{\text{R}}$ simulation that solves only the runner domain using the DNN-based boundary conditions takes ~ 9 min. on the same GPU, i.e. the two tools have a cost ratio of 0.6. The computational budget for any optimization run is set to 150 cost units; this includes the cost for creating the DB$_{\text{DNN}}$, configuring and training the DNNs.

The convergence histories of all optimization runs as well as a close up view to the optimal (re-evaluated on CFD$^{\text{F}}$) solutions are presented in Fig. 6. The optimization on the CFD$^{\text{F}}$ decreased the objective function by $\sim 61\%$. Regarding the runs based on CFD$^{\text{R}}$ which makes use of DB$^{50}_{\text{DNN}}$, DB$^{75}_{\text{DNN}}$ and DB$^{100}_{\text{DNN}}$ a reduction of $\sim 68\%$, $\sim 65\%$ and $\sim 61\%$ is obtained (after re-evaluations on the CFD$^{\text{F}}$ model), respectively.

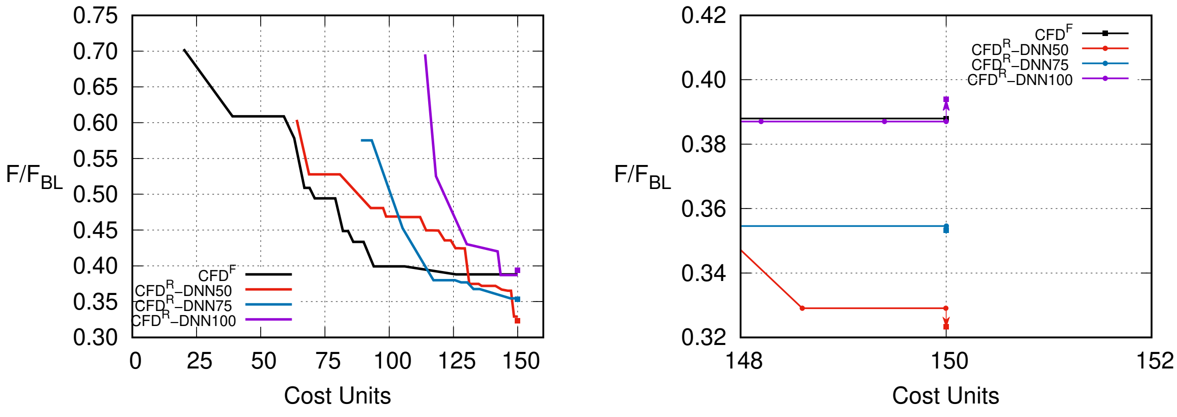

Figure 6: Convergence histories of the MAEA-based optimizations (left); CFD$^{\text{F}}$ (black), CFD$^{\text{R}}$ with DB$^{50}_{\text{DNN}}$ (red), DB$^{75}_{\text{DNN}}$ (blue) and DB$^{100}_{\text{DNN}}$ (purple). Cost units from 1 to the beginning of the colored lines correspond to the evaluation of training patterns, the training of DNNs as well as the cost of the very first MAEA generation. Close-up view (right) of the end of the optimization in which the optimized solutions are re-evaluated on the CFD$^{\text{F}}$ (filled squares).

The (non-dimensionalized) objective and constraint values of the optimized solutions are summarized in Table 2. It can be seen that the optimizations relying on CFD$^{\text{R}}$ with DB$^{50}_{\text{DNN}}$ and DB$^{75}_{\text{DNN}}$ outperform the CFD$^{\text{F}}$-based one. With the decided computational budget, the run with

$\text{DB}^{100}_{\text{DNN}}$ does not have enough cost units left for the optimization, given that 102 cost units have already been spent for populating the DB_{DNN} and training the DNNs. Regarding the cavitation constraint (second row of Table 2), the minimum pressure on the runner blades resulted from all optimization runs is higher compared to that of the baseline geometry, i.e. the cavitation risk is less. As far as the torque constraint is concerned CFD^{R} with $\text{DB}^{75}_{\text{DNN}}$ and $\text{DB}^{100}_{\text{DNN}}$ yield more or less the same torque with the baseline geometry, while CFD^{R} with $\text{DB}^{50}_{\text{DNN}}$ yields more torque.

In all cases studied with the CFD^{R}, the objective function value is computed with adequate accuracy; the percentage errors can be found after re-evaluations on the CFD^{F} are 1.7%, 0.3% and 1.1% for the 50, 75 and 100 training patterns, respectively. The very small error values confirm that the use of an optimization algorithm to select the DNNs' hyperparameters (as described in Sec. 3) is absolutely useful.

	Optimization on CFD^{F}	Optimizations on CFD^{R}		
		$\text{DB}^{50}_{\text{DNN}}$	$\text{DB}^{75}_{\text{DNN}}$	$\text{DB}^{100}_{\text{DNN}}$
Objective	0.388	**0.323**	0.353	0.394
Min. pressure	1.01	**1.24**	1.14	1.14
Torque	1.039	**1.012**	0.994	1.007

Table 2: Comparison of the optimized (within the pre-decided computational budget) solutions resulted from the optimizations on CFD^{F} and CFD^{R} (the tabulated values of the latter have been computed by means of CFD^{F}re-evaluations). All quantities are non-dimensionalized by the corresponding values of the baseline geometry. The objective function (to be minimized) is defined in Sec. 3. The "min. pressure" is a measure for possible cavitation; this should exceed 1. Figures in bold highlight the best values obtained from the CFD^{R}-based runs.

A comparison of the circumferential pressure distribution, used for computing the objective function, between the baseline and the optimized from the CFD^{R} model with $\text{DB}^{50}_{\text{DNN}}$, is presented in Fig. 7. A comparison of the pressure on a constant radius surface used for the definition of the objective function between the baseline and optimized geometries is shown in Fig. 8. Apart from the amplitude itself, the pressure values themselves are also decreased in the optimized geometry.

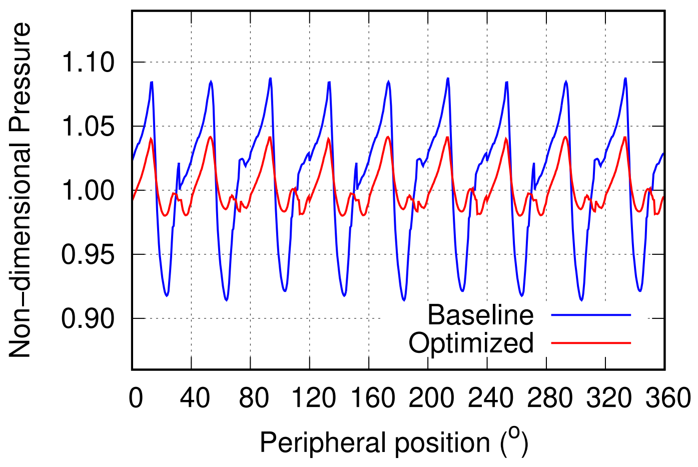

Figure 7: Comparison of the circumferential pressure distribution on the baseline (blue) and the optimized by CFD^{R} with $\text{DB}^{50}_{\text{DNN}}$ (red) geometries. Pressure values are non-dimensionalized with the mean pressure value of the baseline geometry.

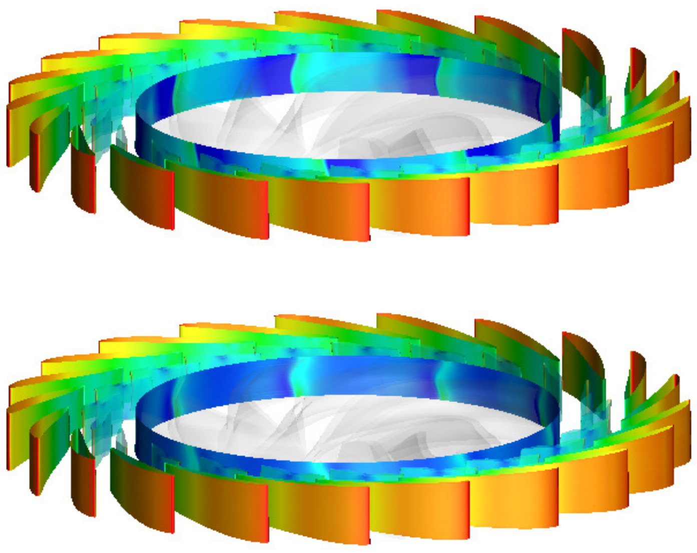

Figure 8: Pressure field on the surface at a constant radius used for the definition of the objective function for the baseline (top) and the optimized (bottom) geometries.

5 CONCLUSIONS

The shape optimization of a hydraulic turbine assisted by a DNN-based surrogate to the mixing plane technique was presented. The aim was to reduce the number of inter-communicating domains for which a CFD solution is necessary and, as a consequence, the computational cost of multi-row CFD simulations. The gain in computational cost was showcased in the shape optimization of the runner blades of a hydraulic turbine where the DNNs were used to "simulate" the presence of the stationary domains namely the stay vanes, the guide vanes and the draft tube. The DNNs were trained on the coordinates of the control points of the NURBS lattice parameterizing the runner blade and deforming the surrounding computational mesh. The averaged quantities exchanged at the interfaces between stationary and rotating domains were predicted by the DNNs and used for simulating the flow only in the runner domain. The optimization studies revealed that the DNN-based surrogate is a reliable tool and when used in the context of an optimization algorithm may lead to better optimized solution (up to 20%), for a given computational cost.

6 ACKNOWLEDGMENTS

This work is supported by the REGALE project (EU Grant Agreement No 956560) co-funded by the General Secretariat for Research and Innovation of the Ministry of Development and Investments under the EuroHPC JU.

REFERENCES

[1] M.G. Kontou, V.G. Asouti, K.C. Giannakoglou, DNN surrogates for turbulence closure in CFD-based shape optimization. *Applied Soft Computing*, **134**, 110013, 2023.

[2] D. Chen, X. Gao, C. Xu, S. Wang, S. Chen, J. Fang, Z. Wang, FlowDNN: a physics-informed deep neural network for fast and accurate flow prediction. *Frontiers of Information Technology & Electronic Engineering*, **23**, 207–219, 2021.

[3] H. Pan, C. Hang, F. Feng, Y. Zheng, F. Li, Improved Neural Network Algorithm Based Flow Characteristic Curve Fitting for Hydraulic Turbines. *Sustainability*, **14**, 10757, 2022

[4] P. Dörfler, M. Sick, A. Coutu, *Flow-induced pulsation and vibration in hydroelectric machinery: Engineer's guidebook for planning, design and troubleshooting*, Springer, London, 2013.

[5] D.H. Kapsoulis, K.T. Tsiakas, X.S. Trompoukis, V.G. Asouti and K.C. Giannakoglou, Evolutionary multi-objective optimization assisted by metamodels, kernel PCA and multi-criteria decision making techniques with applications in aerodynamics. *Applied Soft Computing*, **64**, 1–13, 2018.

[6] V.G. Asouti, X.S Trompoukis, I.C. Kampolis and K.C. Giannakoglou, Unsteady CFD computations using vertex-centered finite volumes for unstructured grids on Graphics Processing Units. *International Journal for Numerical Methods in Fluids*, **67**, 232–246, 2011.

[7] K.C. Giannakoglou, The EASY (Evolutionary Algorithms SYstem) software, `http://velos0.ltt.mech.ntua.gr/EASY`, 2008.

[8] I.C. Kampolis, X.S Trompoukis, V.G. Asouti and K.C. Giannakoglou, CFD-based analysis and two-level aerodynamic optimization on Graphics Processing Units. *Computer Methods in Applied Mechanics and Engineering*, **199**, 712–722, 2010.

[9] P. Spalart, S. Allmaras, A one-equation turbulence model for aerodynamic flows. *Recherche Aerospatiale*, **1**, 5–21, 1994.

[10] X.S. Trompoukis, K.T. Tsiakas, V.G. Asouti, K.C. Giannakoglou, Continuous adjoint-based shape optimization of a turbomachinery stage using a 3D volumetric parameterization. *International Journal for Numerical Methods in Fluids*, doi:10.1002/fld.5187, 2023.

[11] D. Kingma, J. Ba, Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations (ICLR)*, 2015