Proceedings of the Canadian Society for Mechanical Engineering International Congress
32nd Annual Conference of the Computational Fluid Dynamics Society of Canada
Canadian Society of Rheology Symposium
CSME/CFDSC/CSR 2025
May 25–28, 2025, Montréal, Québec, Canada

Current Development Status of a Design Framework for Hydraulic Machines

Alexander Tismer and Stefan Riedelbauch

Institute of Fluid Mechanics and Hydraulic Machinery, University of Stuttgart, Stuttgart, Germany
alexander.tismer@ihs.uni-stuttgart.de

***Abstract*—The design of hydraulic machinery and its components benefits from the increase of computational resources. An application, free of license costs, is presented to optimize machines on large computing clusters. The system is connected to single- and multi-objective Evolutionary Algorithms via a Python interface. A simple demonstration case with online documentation explains the design concepts of the framework. The single-objective flow optimization of a kinetic turbine and the multi-objective flow and structure optimization of an axial turbine show its application to "real-world" problems. An Artificial Intelligence-based multi-objective optimization of an axial runner shows a substantial reduction of the required computing resources.

***Keywords-component*—hydraulic machinery; optimization; artificial intelligence; evolutionary algorithm**

I. INTRODUCTION

Hydraulic machines are increasingly being designed automatically. The greater the available computing resources, the more automated the design process must be. Large computing resources enable to perform thousands of simulations of a hydraulic machine consisting of guide vane, runner and, draft tube. Automated optimizers reduce the manual effort, because the process is able to run without human input.

There are commercial, but only a few academic or license-free turbomachinery design systems. Many frameworks are limited to one special component, so that the entire hydraulic machine cannot be designed as a whole. Additionally, many existing frameworks have fixed parameterizations, that prevent the user from customizing the system.

The academic software environment Computer Aided Design and Optimization (CADO) [1] designs axial and radial turbomachinery. Both types of simulations, Computational Fluid Dynamics (CFD) and Computational Solid Mechanics (CSM), are carried out. The framework uses an elliptic mesh generator to create the block-structured meshes used to predict the flow; unstructured tetrahedral meshes are used for CSM simulations. Differential Evolution (DE) performs a multi-objective optimization in combination with a surrogate model.

The academic- and commercial-driven project BCAM-Baltogar Industrial Platform for Engineering Design (BBIPED) [2] was developed until 2016. The main motivation is to reduce the user's learning curve in order to enable them to implement new open-source software as quickly as possible in the development process. The framework uses an open-source flow solver and meshing application for performing CFD simulations.

The FreeCAD Beltrami Workbench (BW) [3] framework is implemented in an open-source Computer Aided Design (CAD) application. Its main functionality is to design, reverse-engineer, and correct new or existing runner blades. In order to provide learning material a YouTube channel, a manual with tutorials, publicly available source code, and readme files are provided [4].

Besides the mentioned design frameworks, many other commercial software is available to design specific machine parts. Therefore, the design of a hydraulic machine including all machine parts, perform coupled simulations, use large computing resources, and apply new features within the procedure is difficult to set up with the existing tools.

This paper describes the current state of a highly-customizable framework to design hydraulic machinery. Ongoing research activities, explained with examples, are described, too.

Parts of this work are sponsored by the German Research Foundation, Grant/Award Number SPP-2353/501932169 and by the High Performance Computing Center Stuttgart, Grant/Award Number parD-tOOMeetsCanada/44196.

II. FRAMEWORK

The framework Design Tool Object-Oriented (dtOO) is implemented in C++ and includes an interface to Python. It has an object-oriented code structure that is available in GitHub [5]. The system is able to define components using free-form lines, surfaces, and regions by using an underlying CAD library. In order to enable high flexibility with maximum possible accuracy, the system is designed for creating hybrid meshes. Typically, the region surrounding the blades or inside the draft tube is meshed with quadrangles. "Critical" regions are filled with tetrahedrons in combination with prismatic layers to model correctly the near-wall flow. Open Field Operation and Manipulation (OpenFOAM) solves the flow of the prepared case by a direct interface to dtOO. By using source-available, open-source, free, or "anything-else-like" packages in Python, the simulated flow field can be processed and evaluated automatically. The dtOO's interface to Python provides an easy embedding of external tools, packages, libraries, or other user-needed parts.

Usage

The core idea behind dtOO is flexibility. One of its main advantages is to have a framework that is completely adjustable by the user. It is realized by the object-oriented code structure, a plugin interface, and the interface to Python. The authors also understand flexibility in the sense of having a software that is free to use for non-commercial applications. Therefore, only license-free components are included in dtOO.

Besides the source code of dtOO, also a documentation is provided in the GitHub repository [5]. For this publication, a simple demonstration case was added to the documentation [6]. Fig. 1 shows a sketch of the *hydFoilOpt* case. It is a two-dimensional blade with three Degrees Of Freedom (DOFs), namely inlet angle α_1, blade thickness t_{mid}, and outlet angle α_2. This profile is optimized for a design head of 0.8 meters.

The fitness function is an average of head deviation and efficiency. Fig. 2 visualizes the flow for two candidates. The non-optimized solution on the left-hand side shows separation

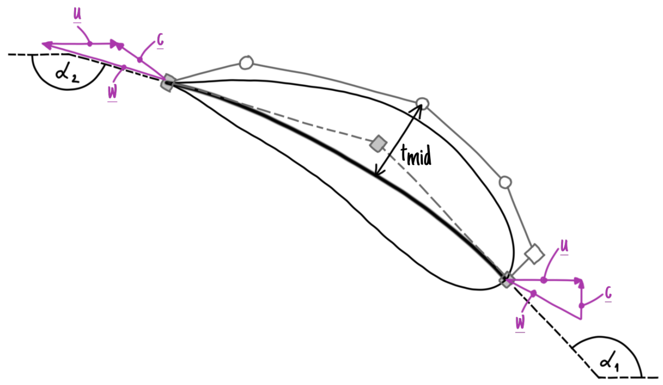

Figure. 1. Sketch of the demonstration case *hydFoilOpt* including geometry, velocity triangles (magenta) at inlet and outlet, and labeled DOFs; construction lines are drawn as gray or thick lines

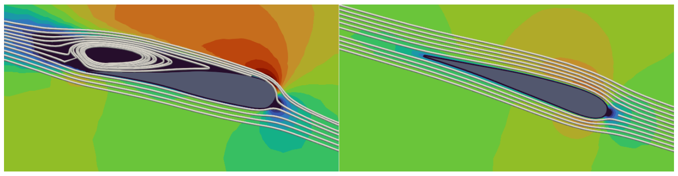

Figure. 2. Non-optimized (left) and optimized candidate (right) of the *hydFoilOpt* case; coloured velocity from 10 (blue) to 30 (red) meters per second; white streamlines indicate a flow separation for the non-optimized candidate

and misaligned flow at the leading edge. The optimized version has smooth streamlines that surround the blade. The main motivation for the new tutorial case is to have a simple case that is optimized with an Evolutionary Algorithm (EA). The structure of the case is the same as for a "real-world" hydraulic machine. Therefore, the case to be optimized can be easily exchanged, as it is only a function for the optimizer.

Within the GitHub repository also Docker container are provided as part of the continuous integration workflow. An interested reader is able to pull and run a dtOO's image by executing the first line of Listing 1. The *hydFoilOpt* case, that is also described at the website, is executed by the >-prefixed commands of Listing 1.

```
docker run -it atismer/dtoo:stable
> cd /dtOO/demo/hydFoilOpt
> export FOAM_SIGFPE=0 && export OSLO_LOCK_PATH=/tmp
> python3 -m doctest build.py
```

Listing. 1. Instructions to download and run a stable container

The interface between the flow problem in dtOO and an optimizer, e.g. *scipy*'s DE algorithm, is shown in Listing 2. The function definition `optimizeHydFoil()` is necessary to have the callback for `differential_evolution()`. The steps for creating the geometry, creating the mesh geometry, perform meshing, perform the simulation, and evaluate the case are implemented in the class methods `Geometry()`, `GeometryMesh()`, `Mesh()`, `Simulate()`, and `Evaluate()`, respectively. All of these functions are documented for the demonstration case to provide an easy start with dtOO.

```
>>> def optimizeHydFoil(x):
...   try:
...     hf = hydFoil(
...       alpha_1=x[0], alpha_2=x[1], t_mid=x[2]
...     )
...     hf.Geometry()
...     hf.GeometryMesh()
...     hf.Mesh()
...     hf.Simulate()
...     fit = hf.Evaluate()
...   except:
...     fit = hydFoil.FailedFitness()
...   return fit
>>> result = differential_evolution(
...   optimizeHydFoil,
...   bounds=[(150., 170.), (155., 175.), (.01, .1),],
...   popsize=3, maxiter=2, polish=False
... )
```

Listing. 2. Python code for the interface between dtOO and *scipy*'s optimizer

Listing 3 shows the definition of the mean line (thick black solid line in Fig. 1) of the *hydFoilOpt* case. First, the imports of dtOO and the predefined builder class are carried out. The builder for the mean line (note: mean plane and mean line is equivalent as the case is two-dimensional) is constructed with the required DOFs. The attribute `alphaOne` expects a scalar function that defines α_1 from hub to shroud. In this two-dimensional case, the angle is constant and, therefore, the function is created as straight line between two points. According to Fig. 1, the second mean line's DOF α_2 is defined in the same way, but not shown in the listing. All other DOFs of the builder `analyticSurface_threePointMeanplaneFromRatio` are set to default values, as they are not used for the demonstration case. The input argument's and return value's type is a bundled class that stores all geometry entities. It is labeled in Listing 3 as `cObj`. Within dtOO many similar builder classes for easy definitions of typical hydraulic components exist, e.g. for hubs, shrouds, thickness distributions, straight draft tubes, and much more.

```
>>> import dtOOPythonSWIG as dtOO
>>> from dtOOPythonApp.builder import (
...   analyticSurface_threePointMeanplaneFromRatio,
...   scaOneD_scaCurve2dOneDPointConstruct
... )
>>> cObj = analyticSurface_threePointMeanplaneFromRatio(
...   "meanplane", spanwiseCuts = [...],
...   alphaOne = scaOneD_scaCurve2dOneDPointConstruct(
...     [
...       dtOO.dtPoint2(0.00, (numpy.pi/180.) * alpha_1),
...       dtOO.dtPoint2(1.00, (numpy.pi/180.) * alpha_1),
...     ],
...     1
...   )(),
...   alphaTwo = [...], ratioX = [...], deltaY = [...],
...   offX = [...], offY = [...],
...   targetLength = [...], targetLengthTolerance = [...],
...   originOnLengthPercent = [...]
... ).buildExtract( cObj )
```

Listing. 3. Meanplane's definition with the usage of a predefined builder class

The subpackage *pyDtOO* provides classes to evaluate flow results that were simulated with OpenFOAM. Listing 4 shows the evaluation for an integration to calculate the total energy at the boundary patch `INLET`. The objects `U` and `p` are created as an instance of `dtValueField` by reading a csv-formatted file. This can also be replaced by a library that directly operates on OpenFOAM data. The methods `IntValueQ()` and `IntMagSquareQ()` provide the value's integration and the squared value's integration times discharge, respectively. Both integrals are necessary to compute the integrated total energy `e` over the boundary surface `INLET`.

```
>>> from pyDtOO import ( dtValueField, dtField )
>>> U = dtValueField( dtField('INLET_U_1000.csv').Read() )
>>> p = dtValueField( dtField('INLET_p_1000.csv').Read() )
>>> e = ( p.IntValueQ() + U.IntMagSquareQ()/2 ) / 9.81
```

Listing. 4. Python classes to support automatic evaluation

III. Examples

The following examples show the flexibility and the application of the framework. The kinetic and the axial turbine emphasize the functionality of dtOO to "real-world" problems. Especially, the kinetic turbine is an example of an uncommon, non-traditional machine. The multi-objective optimization of the axial turbine is an additional example of "real-world" problems, but also serves as a case study for research. Repeated results enable to handle random effects within EAs. The last example clearly shows the flexibility by providing easy extension possibilities and current development directions of dtOO.

A. Power Optimization of a Kinetic Turbine

A kinetic turbine consisting of a guide vane, runner, and a segmented diffuser is parameterized and optimized using dtOO. Fig. 3 shows the turbine's geometry. The outer diameter of the runner and the segmented diffuser's height is set to 4 meters. In order to limit the maximum dimensions of the machine, the diffuser's width is limited to 12 meters. The machine is parameterized with 30 DOFs. For the optimization, the runner and the diffuser are able to vary, the guide vane and hub is fixed.

The complete optimization procedure is implemented in Python using dtOO's and additional libraries. An island-based parallel optimization algorithm [7] is applied to efficiently solve the problem. The optimization is performed on the supercomputer HPE Apollo Hawk. A combination of power output and cavitation tendency serves as fitness function. The optimizer seeks for the minimum of this negated function. Therefore, the lower the fitness, the better the candidate. The power output and cavitation tendency corresponds to the simulated torque, resulting from pressure and shear forces, and the blade's area below vapour pressure, respectively.

The largest optimization runs that use 8192 cores for 24 hours result in fitness values being four times smaller than the initialization. During the largest optimization run 16842 candidates are investigated. In order to sort out candidates with a negative torque, these machines are treated as failed designs. In sum 5597 candidates fail and, therefore, have the highest possible fitness value. It was found that the island model scales well. An over-ideal speedup value of 35% above ideal was observed. The worst speedup value was 28% below ideal for the largest runs. The over-ideal speedup is a strong indication that island models are beneficial for optimization problems in general. The lowest speedup indicates an overall well scaling

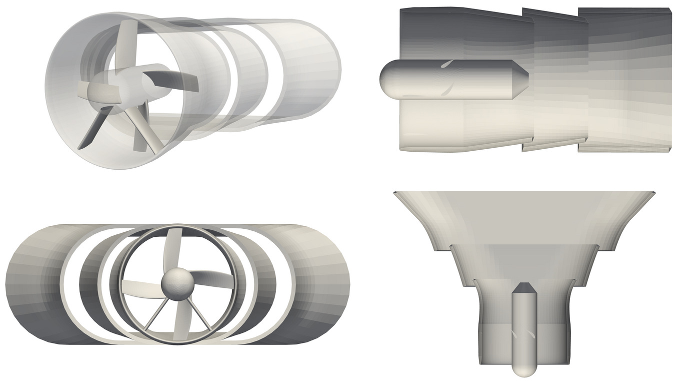

Figure. 3. Three-dimensional view of kinetic turbine with two guide vanes, four runner blades, and a segmented diffuser; the flow enters the machine through the inlet section, but also through the two diffuser's vents

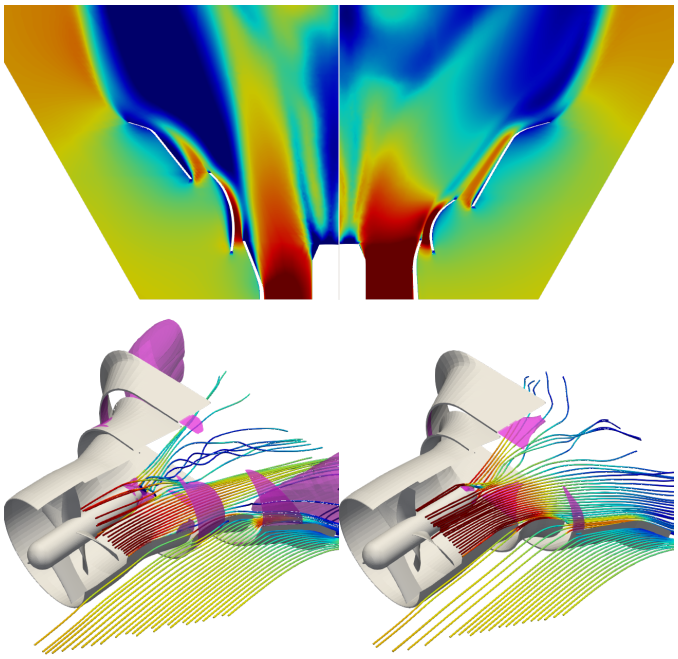
Figure. 4. Optimization's results for kinetic turbine; non-optimized and optimized candidate shown on left-hand and right-hand side, respectively; upper part shows horizontal cut through diffuser with meridional velocities colored from blue to red that corresponds to 0 to 6 meters per second, respectively; lower part contains three-dimensional geometry, streamlines colored by velocity, and backflow regions colored in magenta

of the island model. Nevertheless, it is important to have in mind that the speedup values are influenced by the random nature of EA optimizers.

Fig. 4 presents the difference of a non-optimized (left) and an optimized (right) candidate. The flow through the machine (lower part of figure) visualized as streamlines is characterized by more swirl behind the hub and larger cavitation volumes for the non-optimized candidate. The backflow occurs within, but also behind, the diffuser. Regions with low meridional velocity (blue areas) extend far into the diffuser, see left-upper part of figure. For the optimized candidate, the backflow is shifted outwards of the diffuser. No large areas of deep blue color and no large magenta-colored volumes are observed inside the diffuser in the right-upper and right-lower figure, respectively. The optimized candidate has a power output of approximately 533 kilo watt and a discharge of 70 cubic-meter per second. For a complete description of this optimization including additional results, it is referred to [8].

B. Multi-Objective Optimization of an Axial Turbine

Fig. 5 gives an overview of the hydraulic machine. The cut-view includes the guide vane, runner, draft tube, and tank. The inner and outer diameter of guide vane and runner is 0.45 and 1.6 meters, respectively. All parts are separately meshed with hybrid topologies for the first two and pure hexahedral meshes for the latter two parts. The parts are coupled with mixing planes between guide vane/runner and runner/draft tube and with a general grid interface between draft tube/tank.

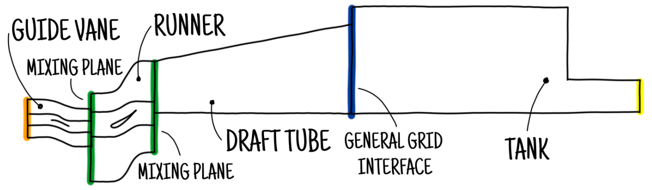

Figure. 5. Cut-view of axial turbine consisting of guide vane, runner, draft tube, and tank from left to right; inlet velocity prescribed at guide vane's inlet (orange); pressure prescribed at tank's outlet (yellow); different machine parts are coupled with mixing plane (green) and general grid interface (blue)

Runner's and guide vane's mean line is defined with six (inlet and outlet angle, mean line's length, mean line's bending position, circumferential and meridional position) DOFs. Each of these DOF is modifiable at three spanwise-constant blade cuts located at hub, mid of hub and shroud, and shroud position. The thickness distribution that is added to the mean line in perpendicular direction is defined with four (blade's thickness at leading edge, in-between and at trailing edge, in-between's position) DOFs at the same spanwise-constant positions. For the blades of the runner, this results in $(6 + 4) \cdot 3 = 30$ DOFs. Due to the constant inflow velocity at the guide vane, only the outlet angles of the guide vane's blades are modifiable. This results in $1 \cdot 3 = 3$ DOFs.

According to Fig. 5, the draft tube is a straight cone. It is defined by its opening angle and its length; resulting in two DOFs. Thus, the whole machine is parameterized with $30 + 3 + 2 = 35$ DOFs.

The machine at hand operates in nominal-load, part-load, and full-load operating condition. The discharge for part-load and full-load corresponds to 87% and 112% related to nominal-load condition of 55.55 cubic-meters per second. The rotation speed is fixed at 150 revolutions per minute for all operating points. For nominal-load the desired design head is prescribed as eight meters. The maximum permissible stress is set to 80 mega pascal.

The fitness function for the multi-objective optimization problem is formulated as an averaged value of the three operating points' fitness f_{pl}, f_{nl}, and f_{fl}. The optimizer minimizes the fitness f according to:

$$\min \left[f = \frac{1}{3} \left(f_{pl} + f_{nl} + f_{fl} \right) \right] \qquad (1)$$

The fitness in nominal-load is an averaged value based on plant efficiency η, cavitation volume V, simulated head H, and maximum von Mises stress σ. It is important to have in mind that plant efficiency η treats all kinetic energy at the draft tube outlet as a loss of the hydraulic machine. In order to have equal scales, for each non-efficiency term a penalty function $F_{\square}()$ with $\square = \{V, H, \sigma\}$ is applied. To put simply: the penalty function maps a cavitation volume, a head, or a maximum stress value to an efficiency-equivalent value. In that sense, the nominal-load fitness

$$f_{nl} = \underbrace{\overbrace{\underbrace{(1-\eta) + F_H(H)}_{\bullet} + F_V(V)}^{\Diamond} + F_\sigma(\sigma)}_{\blacklozenge} \tag{2}$$

and part-load or full-load fitness

$$\left.\begin{matrix} f_{pl} \\ f_{fl} \end{matrix}\right\} = \underbrace{\overbrace{\underbrace{(1-\eta)}_{\bullet} + F_V(V)}^{\Diamond} + F_\sigma(\sigma)}_{\blacklozenge} \ . \tag{3}$$

is defined. The simulated head is only included in equation (2), but not in equation (3). Therefore, the head is only penalized for nominal-load condition. It is clear that the value f_{pl} and f_{fl} is not equal in equation (3), because for each operating point η, V, and σ changes on the right-hand side.

For the axial turbine three optimization runs, with different active terms of equations (2) and (3), are performed. The symbols $\bullet$, $\Diamond$, and $\blacklozenge$ in the equations explain which term is active or not for an optimization. As an example the $\Diamond$-optimization includes the first three and first two terms of equations (2) and (3), respectively. In this case the stress term $F_\sigma(\sigma)$ is neglected. Each setup is repeated 2 times, to minimize random effects from the results. All optimization runs are performed on the supercomputer HPE Apollo Hawk. Fig. 6 displays the results of the parallel optimization runs using the island model. Within the figure, the arrows at the axis point in decreasing fitness and, hence, indicate better results. The left figure shows that optimizing for cavitation ($\Diamond$) or cavitation and stresses ($\blacklozenge$) reduces the plant efficiency in all operating points. According to the right figure, an optimization without considering the stresses ($\bullet$ and $\Diamond$) produces candidates that are not within the allowed stress range (gray-colored area). Furthermore, the middle figure shows that an optimization without including cavitation ($\bullet$), produces candidates with high cavitation in nominal-load and full-load condition. Also illustrated in Fig. 6 is the finding that the spread is small due to random effects. It is clear that in the right figure large spreads occur, but those spreads are observed for values that are not considered within this specific run. For a complete description of this optimization including more results, it is referred to [9].

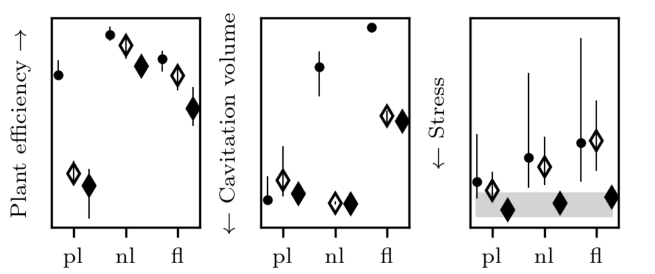

Figure. 6. Multi-objective optimization's results for three different groups of runs; symbols according to equations (2) and (3); marker indicates the mean value; line gives the uncertainty within each setup

C. AI Acceleration

For this Artificial Intelligence (AI)-supported optimization, the runner of the previous optimization is used again. However, instead of a scalar optimizer, a Pareto optimizer is applied to minimize

$$\min \left[\underline{f} = \frac{1}{3} \begin{pmatrix} 3 - \overbrace{(\eta_{nl} + \eta_{pl} + \eta_{fl})}^{\text{Av. efficiency}} \\ 3\,F_H(H_{nl}) \\ F_V\ \underbrace{(V_{nl} + V_{pl} + V_{fl})}_{\text{Av. cavitation volume}} \end{pmatrix} \right] \ . \tag{4}$$

The problem is optimized with the Non-dominated Sorting Genetic Algorithm II (NSGA2) and again with the island model for parallelization. The framework is extended to use AI for accelerating the EA optimization procedure. The main assumption for the following workflow is that EAs decide mainly by a tendency and not an actual difference. In other words: for the algorithm, it is sufficient to know if a candidate is better than another. The exact difference does not have to be known exactly, because the main candidates' selections in an EA are based on "better-or-not" rules.

Fig. 7 explains the core idea of the workflow. It starts with the DOFs on the left side that define the geometry, define the mesh, define the simulation case, define the evaluation, and define Flow Areas (FAs). The FAs that surround the blade are shown as red and gray regions with labels 1 to 8, see Fig. 7. For each FA velocity and pressure values are extracted and processed in an Autoencoder (AE). It extracts features and, hence, performs a dimension reduction (latent space). Then, a Spectral Clustering Algorithm (SCA) clusters the flow fields in the AE's latent space, see right-hand side of Fig. 7. This procedure is performed in all 8 FAs and forms an individual Cluster ID (CI) of 8 digits for each candidate. Based on its CI and the other individuals inside this cluster, a candidate's fitness value is estimated. In order to get a mapping, without the flow solution, between CI and DOFs, for each FA a Neuronal Network (NN) is trained, see curvy gray arrow from left to right at top of Fig. 7. If all AI agents are trained, the described workflow with the NNs provide a fast estimation for the CI and, hence, the fitness values based on the DOFs.

All agents, the AEs, the SCAs, and the NNs, are retrained during an ongoing optimization. In order to save computational

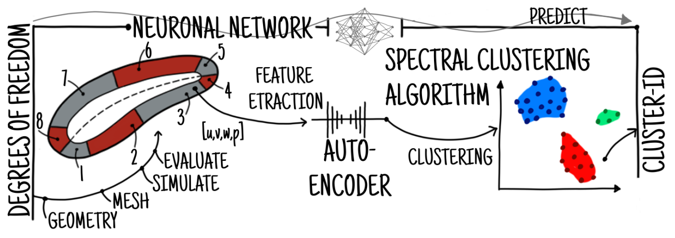

Figure. 7. AI-supported optimization workflow including main components; clustering of the flow fields is achieved by an AE that extracts main features for every FA; candidate's CI is defined by the cluster it belongs to; NN provides a fast candidate's fitness approximation by predicting its CI based on its DOFs

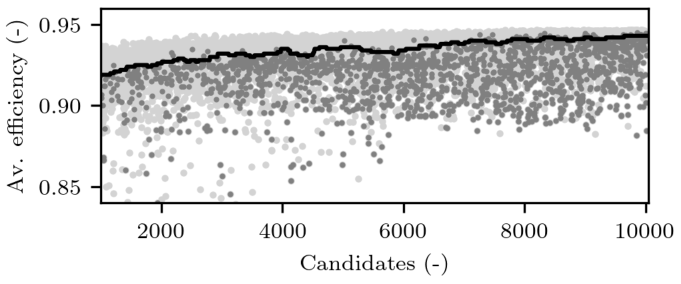

Figure. 8. Averaged fitness during the optimization's course for the runner; dark gray dots represent AI-based estimations for the candidates' fitness; black line is the dynamic barrier for decision "simulate-by-CFD-or-not"

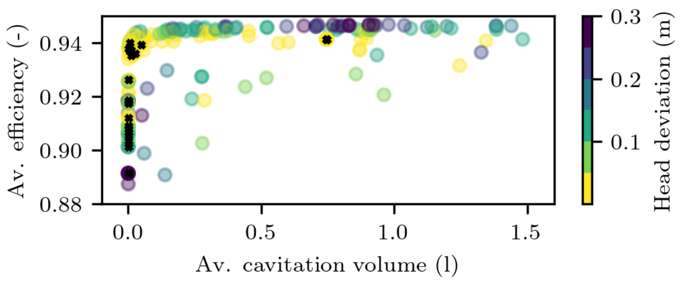

Figure. 9. Pareto front of the NSGA2 run; candidates marked with a black cross symbol are AI-estimated; head deviation is the absolute difference of design and simulated head

time, candidates' DOFs are checked with NNs. If a candidate belongs to clusters with fitness values below a barrier, this candidate is not simulated by CFD. Instead, the AI-based agents predict this candidate's fitness values and, therefore, save computational time. It should be noted that for the decision "simulate-by-CFD-or-not" the averaged efficiency (see underbraced terms in (4)) is used and not the fitness value. To put simply: the decision based on averaged efficiency performs better than on fitness. This is mainly caused by the high fluctuations of the cavitation volume.

As already mentioned the agents are continuously retrained during the optimization. Additionally, the barrier for the decision "simulate-by-CFD-or-not" is dynamic, too. Fig. 8 shows the course of an optimization consisting of 10,000 candidates. It is obvious that many evaluations during the NSGA2 run are far away from optimum result and, as already mentioned, the exact value does not significantly change the optimization's course. The dynamic barrier is based on the mean value of the training samples for the AI agents. On the one hand, the quality of the AI estimation increases due to a larger number of training samples. On the other hand the candidates improve themselves. In that sense, the dynamic barrier approaches larger (better) efficiency values, see Fig. 8.

Overall, the AI-based optimization algorithm estimates 21 percent of the candidates. And, since the computational effort of training the AI-agents is negligible, around 20 percents of computational resources are saved.

Fig. 9 displays the Pareto front for the multi-objective optimization. It is clear that few AI-estimated individuals remain in the Pareto front. Those candidates are predicted at a late optimization stage. Therefore, the estimation has a high quality and does not affect the final results. For a detailed description of the workflow and a deeper investigation of the prediction's influence on the result, please refer to [10].

IV. CONCLUSION

The software dtOO is a highly-customizable framework that is able to optimize "real-world" hydraulic machines. The Python interface provides easy connections to other libraries that support the optimization process. For interested readers the whole framework is available on GitHub. Besides the source code, demonstration cases and documentation is available. The framework is easily extendable, because of its object-oriented structure and the Python interface. A coupling to an island-based parallel optimization algorithm was used to design a kinetic turbine in 24 hours on a supercomputer. The same coupling designs an axial turbine for multiple operating points, too. An AI-based workflow accelerates the NSGA2 optimization of an axial runner by estimating approximately 20 percents of the CFD simulations.

ACKNOWLEDGEMENT

The authors would like to thank Stefan Fraas, Simon Eyselein, Tobias Rentschler and Rohit Raj for their help and support in making this work possible.

REFERENCES

[1] V. Kármán and T. Verstraete, "Cado : a computer aided design and optimization tool for turbomachinery applications," *2nd International Conference on Engineering Optimization (EngOpt 2010), Lisbon, Portugal*, pp. 6–9, 2010.

[2] C. Alonso Montes, I. Garcia-Beristain, A. Ramezani, and L. Remaki, "Bbiped: Bcam-baltogar industrial platform for engineering design," *18th International Conference on Circuits, Systems, Communications and Computers (CSCC 2014), Santorini, Greece*, 07 2014.

[3] SimTurb, C. Hennes, luzpaz, and M. Wilfinger, *Simturb/Beltrami*, 8 2024. [Online]. Available: https://github.com/Simturb/Beltrami

[4] M. Sabourin, "Analytics of turbomachine geometry," Canadian Society for Mechanical Engineering International Congress 2023. Université de Sherbrooke. Faculté de génie, 2023. [Online]. Available: http://dx.doi.org/10.17118/11143/21018

[5] A. Tismer, *ihs-ustutt/dtOO*, 1 2025. [Online]. Available: https://github.com/ihs-ustutt/dtOO

[6] "Demonstration; dtOO documentation — ihs-ustutt.github.io," https://ihs-ustutt.github.io/dtOO/demonstration.html#module-hydFoilOpt.build, [Accessed 11-02-2025].

[7] D. Izzo, M. Ruciński, and F. Biscani, *The Generalized Island Model*, F. Fernández de Vega, J. I. Hidalgo Pérez, and J. Lanchares, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28789-3_7

[8] A. Tismer and S. Riedelbauch, "Optimization of a diffuser augmented kinetic turbine with the island model," *IOP Conference Series: Earth and Environmental Science*, vol. 1411, no. 1, p. 012054, nov 2024. [Online]. Available: https://dx.doi.org/10.1088/1755-1315/1411/1/012054

[9] S. Fraas, A. Tismer, and S. Riedelbauch, "Multidisciplinary optimization of an axial turbine," *IOP Conference Series: Earth and Environmental Science*, vol. 1411, no. 1, p. 012012, nov 2024. [Online]. Available: https://dx.doi.org/10.1088/1755-1315/1411/1/012012

[10] S. Eyselein, A. Tismer, R. Raj, T. Rentschler, and S. Riedelbauch, "Ai-based clustering of numerical flow fields for accelerating the optimization of an axial turbine," *Energies*, vol. 18, no. 3, 2025. [Online]. Available: https://www.mdpi.com/1996-1073/18/3/677