UNSTEADY CFD ANALYSIS AND SHAPE OPTIMIZATION OF A HYDRAULIC TURBINE

V.G. Asouti\textsuperscript{a} - X.S. Trompoukis\textsuperscript{a} - K.C. Giannakoglou\textsuperscript{a} - E.A. Kontoleontos\textsuperscript{b}

\textsuperscript{a}Parallel CFD & Optimization (PCOpt) Unit, School of Mechanical Engineering, National Technical University of Athens (NTUA), Athens, Greece, \{vasouti,xeftro,kgianna\}@mail.ntua.gr

\textsuperscript{b}Andritz Hydro GmbH, Linz, Austria, Evgenia.Kontoleontos@andritz.com

ABSTRACT
This paper presents the CFD-based constrained shape optimization of the runner of a hydraulic turbine for minimum pressure pulsations between the guide vanes and the runner. The GPU-accelerated flow solver PUMA, which solves the (U)RANS equations coupled with the Spalart-Allmaras turbulence model and a free-form deformation tool based on volumetric NURBS are used. A control lattice encapsulates the runner’s blade and the coordinates of (some of) its control points are the design variables; this results in 180 design variables. The optimization is carried out by means of an evolutionary algorithm assisted by surrogate evaluation models and the principal component analysis, in order to minimize the computational cost. During the one- and two-operating point optimizations, all CFD simulations rely on the mixing plane model; however, at the end, the so-optimized solution is re-evaluated using the sliding plane technique.

KEYWORDS
CFD, Design/Optimization, Hydraulic Turbines, Evolutionary Algorithms, Surrogate Evaluation Models

NOMENCLATURE

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EA</td>
<td>Evolutionary Algorithm</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>LCPE</td>
<td>Low Cost Pre-Evaluation</td>
</tr>
<tr>
<td>MAEA</td>
<td>Metamodel-Assisted EA</td>
</tr>
<tr>
<td>NURBS</td>
<td>Non-Uniform Rational B-Splines</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>RANS</td>
<td>Reynolds-Averaged Navier-Stokes</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Functions</td>
</tr>
<tr>
<td>URANS</td>
<td>Unsteady RANS</td>
</tr>
</tbody>
</table>

Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>artificial compressibility coefficient</td>
</tr>
<tr>
<td>$\delta_{km}$</td>
<td>Kronecker symbol</td>
</tr>
<tr>
<td>$\epsilon_{ikl}$</td>
<td>permutation symbol</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>offspring population (in EA)</td>
</tr>
<tr>
<td>$\mu$</td>
<td>parent population (in EA)</td>
</tr>
<tr>
<td>$\nu$, $\nu_t$</td>
<td>bulk and turbulent kinematic viscosities</td>
</tr>
<tr>
<td>$\tau$, $t$</td>
<td>pseudo and real time</td>
</tr>
<tr>
<td>$\tau_{km}$</td>
<td>stress tensor</td>
</tr>
<tr>
<td>$\omega_l$</td>
<td>angular velocity</td>
</tr>
<tr>
<td>$p$</td>
<td>static pressure</td>
</tr>
<tr>
<td>$u_k$</td>
<td>peripheral velocity, $u_k = \epsilon_{klm} \omega_l \epsilon_{lmn}$</td>
</tr>
<tr>
<td>$v_k$</td>
<td>absolute velocity, $v_k = w_k + u_k$</td>
</tr>
<tr>
<td>$w_k$</td>
<td>relative velocity</td>
</tr>
<tr>
<td>$x_k$</td>
<td>Cartesian coordinates</td>
</tr>
<tr>
<td>$\mathcal{P}$, $\mathcal{D}$</td>
<td>turbulent production and destruction</td>
</tr>
<tr>
<td>$S$</td>
<td>population size</td>
</tr>
<tr>
<td>$U$</td>
<td>vector of flow variables</td>
</tr>
</tbody>
</table>
INTRODUCTION

During the last decades, the international market competition as well as the technological advancements resulted in more compact designs, i.e. smaller machines. At the same time, the increased energy demands and grid regulation services led the hydropower suppliers to operate the hydraulic turbines at their operating limits, in a far more dynamic way, at flow conditions not experienced in the past. At these conditions, dynamic phenomena, such as pressure pulsations in the fluid flow through the turbine, regularly appear, Dörfler et al. (2013). Such phenomena are decisive for the smooth and safe operation of hydraulic machines. Thus, attention was paid to fluctuating loads and the unsteady operational behaviour, Gentner et al. (2014).

Next to flow simulations, engineering research focuses on the design of new, optimized, hydraulic components by considering pressure pulsations and unsteady phenomena. This work is concerned with the CFD-based shape optimization of a hydraulic turbine where pressure pulsations between the runner and, additionally, the guide vanes need to be minimized, without damaging the turbine head or the efficiency while ensuring that no cavitation occurs in the runner blades. Single- and two-point optimization studies are performed. The flow analysis is carried out using the a GPU-accelerated flow solver of the (U)RANS equations coupled with the Spalart-Allmaras turbulence model, Asouti et al. (2011).

The simulation involves the following components of the turbine: stay vanes, guide vanes, runner blades and draft tube. The interfaces between guide vanes and runner as well as runner and draft tube can be modelled either via the mixing or the (more expensive) sliding plane technique, by performing a steady or unsteady flow simulation, respectively. For the shape optimization, only the runner’s blade shape is allowed to vary. To do so, the blade geometry is parameterized using a free-form deformation tool, based on volumetric NURBS, customized to turbomachinery bladings.

Optimization is carried out by means of an EA assisted by surrogate evaluation models or metamodels and the principal component analysis (PCA), Kapsoulis et al. (2018). Online-trained radial basis function (RBF) networks are used as metamodels, to avoid excessive usage of the expensive CFD tool. The PCA is used to tackle the so-called “curse of dimensionality”, as the parameterization of the runner geometry introduces a great number of design variables. In specific, the PCA is implemented during the application of the evolution operators and/or the metamodels’ training, in order to enhance the exploration and exploitation capabilities of the former and get more accurate predictions by the latter. The optimization runs are performed on an HPC cluster with GPUs. During the optimization, the steady-state variant (with the mixing plane technique) of the CFD code is used so as to find the optimized geometry at a reasonable cost; the flow within the so-optimized geometry is simulated again using the sliding plane technique and compared with the baseline as well as the outcome of the steady flow simulation.

METHODS AND TOOLS

This section describes the background methods and tools used for the shape optimization of the hydraulic turbine.

Flow Model - The CFD Tool

The CFD code (PUMA s/w by NTUA, Kampolis et al. (2010); Asouti et al. (2011); Trompoukis et al. (2021)) solves the (U)RANS equations for incompressible fluids on GPUs, on unstructured/hybrid meshes, using the vertex-centered finite volume approach. The artificial com-
pressibility method is used. The URANS equations are written as

\[ \mathcal{M}_{nm} \frac{\partial U_m}{\partial t} + \Gamma_{nm} \frac{\partial U_m}{\partial \tau} + \frac{\partial f_{nk}^{\text{inv}}}{\partial x_k} - \frac{\partial f_{nk}^{\text{vis}}}{\partial x_k} + S_n = 0, \quad n \in [1, 4] \quad \text{and} \quad k \in [1, 3] \quad (1) \]

where

\[ \mathcal{M} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \Gamma^{-1} = \begin{bmatrix} \frac{1}{\nu} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad f_{k}^{\text{inv}} = \begin{bmatrix} w_k \\ w_k v_1 + p \delta_{1k} \\ w_k v_2 + p \delta_{2k} \\ w_k v_3 + p \delta_{3k} \end{bmatrix}, \quad f_{k}^{\text{vis}} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \end{bmatrix}, \quad S = \begin{bmatrix} 0 \\ \varepsilon_{1k} \dot{\omega} \dot{v}_k \\ \varepsilon_{2k} \dot{\omega} \dot{v}_k \\ \varepsilon_{3k} \dot{\omega} \dot{v}_k \end{bmatrix} \]

and \( \beta \) is a user-defined parameter corresponding to an artificial speed of sound. \( \mathbf{U} = [p, v_1, v_2, v_3]^T \) is the flow variables vector and \( f_{k}^{\text{inv}}, f_{k}^{\text{vis}}, S \) are the inviscid, viscous/turbulent fluxes and the source terms, respectively. The stress tensor is defined as \( \tau_{km} = (\nu + \nu_t) \left( \frac{\partial v_k}{\partial x_m} + \frac{\partial v_m}{\partial x_k} \right) \).

Turbulence is modeled via the one-equation Spalart–Allmaras model,

\[ \frac{\partial \tilde{v}}{\partial t} + \frac{\partial \tilde{v}}{\partial \tau} + \frac{\partial}{\partial x_k} (w_k \tilde{v}) - \frac{1}{\sigma} \frac{\partial}{\partial x_k} \left[ (\nu + (1 + C_{h2}) \tilde{v}) \frac{\partial \tilde{v}}{\partial x_k} \right] + \frac{C_{h2} \tilde{v}}{\sigma} \frac{\partial}{\partial x_k} \left( \frac{\partial \tilde{v}}{\partial x_k} \right) - \mathcal{P} + \mathcal{D} = 0 \quad (2) \]

where \( \mathcal{P} = C_{bi} (1 - f_{t2}) \tilde{S} \tilde{v} \) and \( \mathcal{D} = (C_{w1} f_w - \frac{C_{w1}}{C_{bi}^2} f_{t2}) (\tilde{v}^2) \) are the production and destruction of model’s variable \((\tilde{v})\). \( \nu_t \) results from \( \nu_t = \nu f_{t1} \) and the constants \( \kappa, C_{bi}, C_{h2}, C_{w1} \) along with the expressions of \( f_{t1}, f_w, f_{t2} \) and \( \tilde{S} \) are given in Spalart and Allmaras (1994).

In steady–state flow simulations, the physical time derivatives vanish and the interaction between adjacent rotating and stationary domains is modeled using the mixing plane technique; according to this, the spanwise distribution of averaged flow variables is communicated between adjacent domains. At each iteration, PUMA computes the circumferentially area–averaged fluxes on the interface between the rotating and stationary domain, based on the current flow solution. Then, based on these fluxes, the spanwise distribution of the mixed-out flow solution, Wang (2014), is computed and communicated to the adjacent domain, where these quantities are imposed as boundary conditions. Main features of the mixing plane technique are that there is no need of a main flow direction (allowing for flow recirculations through the interface), and that fluxes crossing the interface are conserved, upon convergence.

The URANS equations are solved using the so-called dual time stepping approach, i.e. the system is linearized and then integrated in time \((t)\), by performing intermediate pseudo-time \((\tau)\) steps. The unsteady interaction between adjacent rotating and stationary domains is modeled via the sliding plane technique. In specific, at the beginning of each real time step, an interface mesh is generated by superimposing the boundary meshes of the two adjacent domains. During the integration in pseudo-time, the flow variables are transferred to the common interface mesh and used to compute fluxes. These are, then, transferred back to both domains, in a conservative manner.

PUMA runs on GPU clusters employing the MPI protocol for data transfer between GPUs on different nodes and the shared, on-node, memory to do the same between on-node GPUs. High parallel efficiency is achieved by overlapping data communications with computations and the use of the so-called mixed precision arithmetic. According to the latter, during the solution of the governing equations, the memory demanding left-hand-side coefficients are computed with double and stored with single precision. This significantly reduces memory footprint and increases parallel efficiency by reducing the number of transactions between GPU’s memory and running threads.
Shape Parameterization and Mesh Deformation

In this work, a free-form deformation technique based on volumetric NURBS, tailored to peripheral blade rows, is used, Trompoukis et al. (2023). This not only controls the shape under consideration but also deforms the volume mesh.

The important feature of this tool is that it perfectly meets the needs of turbomachinery applications since it ensures hub and shroud axisymmetry and takes periodicity into account. This is achieved through an intermediate coordinate system transformation from the Cartesian into a \((\eta, \theta, \xi)\) system. Let \(C^H\), \(C^S\) be functions of the normalized streamwise parameter \(\xi\) describing the generatrices of hub and shroud surfaces in the meridional plane, so that, the image of any point along the hub \((r^H)\) and shroud \((r^S)\) on the meridional plane is given as \(r^H = C^H(\xi)\) and \(r^S = C^S(\xi)\), respectively. The image \((r)\) of any point between the hub and shroud is expressed in terms of the spanwise parameter \(\eta\) as \(r(\eta) = (1 - \eta)r^H + \eta r^S\). Figure 1 presents the runner geometry in the Cartesian (left), meridional (center) and \((\eta, \theta, \xi)\) transformed (right) coordinate system. One may notice that the runner is practically transformed into a linear cascade. It is important that the lattice that controls both the runner’s shape and its surrounding mesh is first defined in the \((\eta, \theta, \xi)\) coordinate system and, then, transformed into the Cartesian one.

![Figure 1: Runner blade geometry in the Cartesian (left), meridional (center) and transformed (right) coordinate system with the lattice of control points used to parameterize its shape.](image)

The EASY Optimization Platform

The Evolutionary Algorithms SYstem (EASY) platform developed by the PCOpt/NTUA is used for the shape optimization, Giannakoglou (2008). EASY implements a \((\mu, \lambda)\) MAEA; in each generation \((g)\) handles and updates (using evolution operators such as parent selection, crossover, mutation, elitism) three populations namely, the offspring population \((S^g_\lambda\) with \(\lambda\) individuals), the parent population \((S^g_\mu\) with \(\mu\) individuals) and the set of the best so-far (elite) individuals \((S^g_e)\).

To keep the optimization cost as low as possible, EASY extensively makes use of surrogate evaluation models or metamodels. In specific, on-line trained metamodels are implemented during the low-cost pre-evaluation (LCPE) phase of the MAEA. Term “on-line trained” is used to distinguish the way EASY is building surrogate evaluation models. In contrast to most other MAEAs, in EASY there is no need for a preliminary sampling of the design space, to collect training patterns and, finally, train the metamodel on them. In EASY, the algorithm starts as a standard EA (in the sense that all \(\lambda\) offspring per generation are evaluated on the high-fidelity code) till the moment a user–defined minimum number \((T^{MM})\) of individuals evaluated on the CFD code are stored in its database; this database can be seen as the pool of training patterns to be selectively used to build personalized metamodels for the offspring in all subsequent
generations. At this point, the LCPE phase starts; during this second phase, for each and every new offspring, a personalized metamodel of local validity is trained on the “closest” (distances measured in the design space) already evaluated individuals (selected from the database) and only the few most promising \( (\lambda_e) \) of them are re-evaluated on the CFD code and added to the database. Without loss of generality, RBF networks are used as metamodels.

In order to avoid the MAEA performance degradation in problems with many design variables, such as the one tackled in this work, the Kernel PCA, Kapsoulis et al. (2018), is additionally employed. The PCA of the current offspring population is used to control the evolution operators and/or reduce the number of input units of metamodels. Regarding the former, the parent population members are transformed into a new feature space (with ordered variances), before applying crossover and mutation. After the application of the evolution operators, the new offspring population is transformed back into the design space. Regarding the use of the PCA in metamodels, the outcome of the same PCA is additionally used for pruning the number of inputs to the RBF networks. For each population member, once the training patterns for its “personalized” metamodel have been selected, these are transformed into the feature space and the metamodel inputs along the direction with the smaller variances are truncated. Thus, the metamodel is built and trained with less sensory units (inputs), i.e. noisy inputs are not taken into account and, thus, more accurate predictions are expected.

**OPTIMIZATION OF THE HYDRAULIC TURBINE**

**Problem Set-up**

The hydraulic turbine to be optimized consists of 21 stay vanes, 21 guide vanes, 9 runner blades and the draft tube (figure 2). Given the above, the computational domain includes 1/3 of the stationary and rotating blades (forming a 120\(^\circ\) sector) and the draft tube; this results in an unstructured mesh of about 5M nodes. It was decided to keep the flow domain that big, despite the use of a steady flow solver with the mixing plane technique, to be ready to perform re-evaluations on the URANS CFD solver with the sliding plane technique. Even if this makes the cost per generation higher, it does not change the message this paper communicates to the reader. The automatic wall treatment based on the unified wall law of Spalding is employed and, as said before, during the optimization, the mixing plane technique is used to account for the interaction of the runner with the guide vanes and the draft tube during the optimization runs.

The runner speed is 1150 rpm. A fixed mass flow rate is imposed at the inlet and a constant pressure at the draft tube outlet. Two operating points (OP1 and OP2) are considered by regulating the mass flow rate through the guide vanes. OP1 and OP2 correspond to 20\(^\circ\) and 10\(^\circ\) guide vane opening angles, respectively. Both points correspond to the same head value (rated head) with OP1 being at \( Q_{\text{max}} \) (full load) and OP2 at \( Q_{\text{min}} \) (part load), in order to cover the full discharge operating range. The objective function \( f(M) \) that quantifies pressure pulsations, for each operating point, is the amplitude of the pressure field computed along a circumference located between the guide vane-runner interface and the runner leading edge (at a specified radial and axial position). Single- (at OP1) and two-point optimization runs are performed. In the second case, the objective function is the weighted sum of the the quantity of interest at the two points, in specific

\[
F_M = 0.7f_M^{(\text{OP1})} + 0.3f_M^{(\text{OP2})}
\]

During the optimization, only the runner blade shape is allowed to change. This is controlled
by the parameterization tool described above, using the $11 \times 3 \times 5$ control lattice of figure 1. 9 out of the 11 series of control points in the streamwise direction are allowed to vary. Active control points can be displaced in the streamwise and pitchwise directions resulting in 180 design variables in total. Control points on the periodic boundaries undergo periodic changes while those lying on the hub or shroud surfaces remain on them since spanwise modifications are not allowed.

All cases were optimized using a MAEA with $(\mu, \lambda) = (10, 18)$. The metamodels were activated after having evaluated the first $T^{MM} = 50$ individuals on the CFD code; during the LCPE phase, $\lambda_e \in [2, 4]$ of them were re-evaluated on the PUMA code. According to the latter, the two top (according to the evaluation on the metamodels) individuals were surely re-evaluated on the PUMA code; then, depending on constraints’ satisfaction and the accuracy of the metamodels’ predictions for the two just evaluated individuals, up to two other individuals were re-evaluated, too. The PCA was activated after the 2nd generation and the metamodels were trained using 40 inputs each (the 40 first principal components, identified by the PCA of the current offspring population). For the sake of comparison, an additional run using a standard $(\mu, \lambda)$ EA was carried out for the single-point optimization case. The computational budget of the optimization was set to 150 evaluations on the PUMA code for the single-point or 400 for the two-point. For the latter, this corresponds to the simulation of 200 candidate solutions since each of them requires two calls to the PUMA code. The computational cost of the steady-state variant of the CFD code was $\sim 15$ min. on a single A100 NVIDIA GPU and the search was parallelized on one computational node with 4 GPUs. This practically allows for 4 concurrent evaluations in the single- or 2 for the two-point optimization.
Optimization Results

The optimization runs aimed at minimizing the pressure amplitude \( f_M^{(\text{OP1})} \) for the single-point and \( F_M \) for the two-point, eq. 3) with constraints on the head value, the turbine efficiency and the cavitation risk. In specific, the percentage error of the head is not allowed to exceed 1.5\%, the turbine efficiency should not be reduced and the minimum pressure computed over the runner blade should be higher than that on the runner of the baseline geometry. The latter is a rather conservative criterion to ensure that no cavitation takes place, given that the baseline configuration is cavitation free.

For the single-point optimization, the comparison of the convergence history of the two runs is shown in figure 3, left. It is obvious that the MAEA outperforms the EA, yielding better objective function value, for the same computational budget. The PCA contributed to the metamodels training since only 40 inputs (the most important ones), much less than the 180 design variables, were used. Due to its double cost, the two-point optimization was performed only with the MAEA; the corresponding convergence history is shown in figure 3, right.

The runner geometry obtained from the single-point optimization (at OP1) was additionally evaluated at OP2 and compared with the outcome of the two-point optimization run. The results are summarized in table 1. The single-point optimization improved the objective at OP1 by 43\% compared to the baseline one, instead of 35\% gain computed by the two-point optimization. On the other side, the optimized runner geometry from the two-point run yielded a better value of the objective at OP2 since this was included in the objective function. Both optimized geometries (shown in figure 4) respect the head and efficiency constraints. However, the geometry from the single-point optimization does not respect the cavitation constraint at OP2; in fact, it yields a lower value for the minimum pressure over the runner blade compared to the baseline geometry. This, practically, signifies the importance of simultaneously handling more than one operating points during the optimization loop, despite the higher cost. As far as the optimized shapes (from either run) are concerned, these change the blade curvature on the hub and shroud surfaces as well as the trailing edge. Comparison of the circumferential pressure distribution

Figure 3: Left: Comparison of the convergence history of the single-point optimization runs performed using the (standard) EA and the MAEA. The objective function value is non-dimensionalized with that of the baseline geometry at OP1. Right: Convergence history of the two-point optimization run. The weighted (as in eq. 3) objective function value is non-dimensionalized in a similar manner.
at OP1, used for computing the objective function $f_M$, between the baseline and the optimized geometries, is presented in figure 5.

<table>
<thead>
<tr>
<th></th>
<th>Single-point OP1</th>
<th>OP2</th>
<th>Two-point OP1</th>
<th>OP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-dim $f_M^{(OP)}$</td>
<td>0.57</td>
<td>0.83</td>
<td>0.65</td>
<td>0.71</td>
</tr>
<tr>
<td>Head error (%)</td>
<td>0.40</td>
<td>1.01</td>
<td>0.46</td>
<td>0.09</td>
</tr>
<tr>
<td>Cavitation criterion</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Efficiency constraint</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1: Comparison of the optimized solutions resulted from the single- and two-point optimization run. The values at OP2 from the single-point run are obtained by re-evaluating the optimized solution at OP2 conditions.

Figure 4: View of the baseline (blue) and the optimized (red), runner blades resulting from the single- (left) and the two-point (right) optimization run.

Figure 5: Comparison of the circumferential pressure distribution on the baseline (blue) and the optimized from the single- (red) and the two-point run (orange) geometries. Pressure values are non-dimensionalized with the mean pressure of the baseline geometry at OP1.
Unsteady Analysis

The optimized geometry resulted from the two-point optimization run was, then, re-evaluated on the URANS flow model, using the PUMA code with the sliding plane technique. The (real) time step size was computed by dividing a full runner rotation into 360 equal steps; this was proved to be sufficient to resolve the flow structures of interest. For the unsteady analysis, the pressure pulsations at each operating point \((j, j=1, 2)\) were quantified using the following expression

\[
f_{SP}^{(Op_j)} = \frac{1}{M_P} \sum_{i=1}^{M_P} \Delta p_i
\]

where \(M_P\) a number of user-defined monitoring points (the same for all operating points) and \(\Delta p_i\) the pressure amplitude computed by post-processing the pressure time-series at point \(i\). It was decided that two monitoring points \((M_P = 2)\) at the same radius and iso-stream position as in the steady-state case (at different angular positions, though) are enough to quantify the pressure pulsations. The solution field of the unsteady simulation on the baseline geometry after 20 periods was used to initialize the run on the new geometry which lasted 6 more periods.

According to the unsteady simulation, the turbine with the optimized runner yields 47\% reduction in \(f_{SP}^{(OP1)}\) and 12\% in \(f_{SP}^{(OP2)}\) compared to the baseline geometry, by respecting the imposed constraints at both operating points. Compared to the optimization with the steady-state CFD code, the reduction in \(f_{SP}^{(Op_j)}\) is greater for OP1 and smaller for OP2. This can be attributed to the fact that the optimization on the steady-state code tried to minimize \(F_M\) with a higher weight for OP1. The time-averaged pressure field in the area between the guide vanes and the runner at OP1 is shown in figure 6; the decrease in the mean pressure value can readily be seen.

![Figure 6: Time-averaged pressure field computed for the baseline (left) and the optimized (right) geometries, at OP1.](image-url)
CONCLUSIONS

The single- and two-point shape optimization of the runner blade of a hydraulic turbine was presented. The optimization target was the minimization of pressure pulsations in the flow through the turbine with head, efficiency and cavitation related constraints.

The runner blade was parameterized with a free-form deformation technique based on volumetric NURBS which resulted in 180 design variables in total. With such a high number of design variables, the use of an EA to perform the optimization is normally prohibitive, due to the high computation cost. Nevertheless, the use of a MAEA that also employs the PCA of the offspring population in each generation, to assist both the application of the evolution operators and the metamodels’ training, proved to be able to solve the problem with an affordable number of CFD evaluations and, thus, at affordable cost.

In order to further speed-up the optimization turnaround time, a steady-state CFD code (with the mixing plane technique) was used as the evaluation tool during the search by the MAEA. The comparison of the optimized geometries resulted from the single- and two-point runs showed that the single-point optimization improves a lot performance at this operating point but fails to meet the cavitation constraint at the second operating point. This, practically proves the importance of a many-point optimization run. The optimized geometry from the two-point run was, then, simulated anew using with the unsteady variant of the CFD code that uses the sliding plane technique. Though the improvements from the unsteady analysis are different from those of the steady-state simulation, the optimized runner geometry constitutes a real improvement of the baseline one.

ACKNOWLEDGEMENT

This work was supported from the REGALE project (EU Grant Agreement No 956560) cofunded by the General Secretariat for Research and Innovation of the Ministry of Development and Investments under the EuroHPC JU. All simulations and optimization studies presented in this work are performed in the framework of Pilot 1 of the REGALE project.

REFERENCES


