7th GRACM International Congress on Computational Mechanics
Athens, 30 June – 2 July 2011

DESIGN OF A HYDROMATRIX TURBINE RUNNER USING AN ASYNCHRONOUS EVOLUTIONARY ALGORITHM ON A MULTI-PROCESSOR PLATFORM

Irida A. Skouteropoulou[1], Stylianos A. Kyriacou [1,2], Varvara G. Asouti[1], Kyriakos C. Giannakoglou[1], Simon Weissenberger[2], Peter Grafenberger[2]

[1]National Technical University of Athens, Parallel CFD & Optimization Unit, Athens, Greece
Correspondance: kgianna@central.ntua.gr

[2]Andriz HYDRO, RD, Lunzerstrasse 78, 4031 Linz, Austria.

Keywords: multi-objective constrained design-optimization, asynchronous evolutionary algorithm, parallelization, hydraulic machines.

Abstract. *The shape optimization of a Hydromatrix® turbine runner using an asynchronous metamodel-assisted evolutionary algorithm is presented. The optimization problem is subject to constraints and is computationally demanding, since candidate runner geometries are evaluated by means of calls to a CFD code. The use of an asynchronous, rather than a conventional (synchronous or generation-based) evolutionary algorithm, aims at maximizing the parallel performance of the design process on any system of interconnected (likely, heterogeneous) processors and minimizing the turnaround optimization time. On the other hand, the use of metamodels contributes to a noticeable reduction of the computational cost, since the costly CFD evaluations are restricted only to promising solutions. The design is performed at three operating points (the best efficiency point, one part- and one full-load points); the quality of candidate runner shapes is quantified by post-processing the computed pressure coefficient distribution over the blades, the computed outlet mass flow and swirl profiles and also in terms of the cavitation index, resulting thus to three objective functions to be minimized. Results are presented in the form of a Pareto front in the 3D objective function space.*

1 INTRODUCTION

Hydromatrix® is an innovative solution for the development of low head hydropower sites [1]. It makes use of a number of relatively small, axial flow, fixed blade type turbine generator units, comprising a factory assembled grid or “matrix”. The individual turbine generator units (also known as “modules”) are designed for collaborative operation, with great flexibility in their setting. The most common ways of installing matrix turbines is in one, two or three rows (Figure 1). The number of matrix turbines and their arrangement in rows depends on the existing civil structure and its position relative to the head- and tail-water elevations.

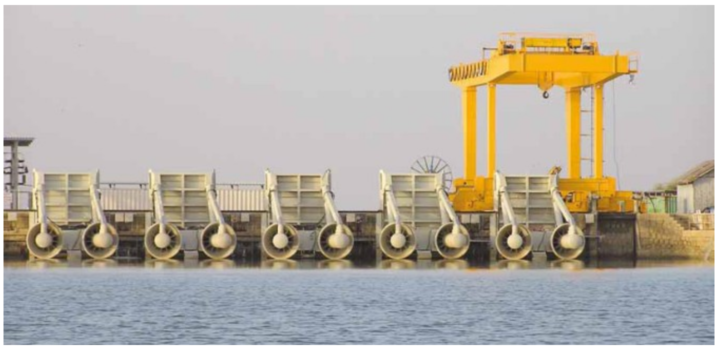
Figure 1. Hydromatrix®: arrangement of matrix turbines in a dam.

The lifting or removal of modules from their operating position, similarly to a sliding gate (Figure 2), enables the passage of flood water and simplifies the inspection and maintenance of matrix turbines. Matrix turbines can easily be integrated into an existing dam and gate structures as well as in greenfield projects. Their most important advantages, compared to conventional low-head solutions, are:

- The required civil construction works are minimal and so does the hydroplant total cost.
- Both geological and hydrological risks (flooding during construction) are kept as small as possible.
- Project schedules, construction and installation time are minimal.

- Inflict on the environment is minimal, too.

Figure 2. Hydromatrix®: in case of flood conditions, inspection or maintenance, some or all of the modules can easily be removed using a small crane.

This paper deals with the design of a Hydromatrix® turbine runner using an asynchronous metamodel-assisted evolutionary algorithm (AMAEA) [2, 3] based on a CFD code as evaluation software. The optimization is carried out on a multi-processor platform, where the use of any generation-based EA makes several processors to remain idle, at the end of each generation, for the purpose of synchronization, waiting for the last evaluation(s) to be completed on the last working processor(s). The use of an asynchronous EA overcomes the synchronization barrier at the end of each generation (a notion which, however, doesn't exist anymore), maximizing thus the parallel efficiency by fully exploiting all available processors. Over and above, since in either synchronous or asynchronous EAs, the optimization turnaround time may increase a lot if the problem-specific evaluation tool (CFD code) is costly, low-cost surrogate evaluation models (or metamodels) are used [3, 4, 5]. In specific, the asynchronous EA is further enhanced by on-line trained radial basis function networks (acting as metamodels) so as to reduce the overall CPU cost of the optimization. As it will be explained in detail below, once a CPU becomes idle, instead of a single individual, a number of trial individuals are generated. All these trial individuals are inexactly pre-evaluated on properly trained local metamodels and the "best" among them, according to the metamodels, undergoes evaluation on the CFD tool running on the idle processor. The computational cost for generating and approximating trial individuals is negligible, compared to the cost of a CFD-based evaluation. This enables a design-optimization procedure with affordable CPU cost and, practically, the maximum possible parallel speed-up (almost 100%, at least theoretically).

The optimization problem is handled as a three objective one and the outcome of the optimization is a front of non-dominated solutions (i.e. a Pareto front), forming a surface in the 3D objective function space. Solutions selected among the members of the computed Pareto front are further analyzed and compared.

2 THE ASYNCHRONOUS METAMODEL-ASSISTED EVOLUTIONARY ALGORITHM

The basic features of AMAEA [2, 3] used herein, including comments on how surrogate evaluation models are implemented and parallelization issues are briefly discussed in the following sections. In what follows, only the variant solving multi-objective (with M functions to be minimized) optimization problems by computing Pareto fronts of non-dominated solutions is presented; from this description, the reader can readily extract the single-objective AMAEA.

2.1 Basic Features

The asynchronous EA [2] this paper relies on utilises a number of search agents located at the nodes of a 2D structured mesh. This is referred to as the "supporting mesh" with dimensions $n_1 \times n_2$, where n_1 and n_2 are user defined integers (Figure 3). The mesh is divided into a number ($n_1 n_2 / 4$) of demes with six nodes each; each deme comprises five search agents and a single pole. It overlaps with four adjacent ones and this overlapping enables the inter-deme exchange of genetic material. To regularize the communication between demes, the supporting mesh is considered to be periodic along its two pairs of opposite sites; for instance, the top-left deme, apart from its two physically adjacent demes, communicates also with the top-right and bottom-left demes of the supporting mesh.

The role of each pole is to store the best-so-far individual (local best) computed by the search agents of its deme and, through them, to affect the formation of new candidate solutions. On the other hand, search agents:

a) generate (based on evolution principles and the relative operators) new candidate solutions,
b) organize evaluation on the problem-specific evaluation tool on idle CPUs and
c) provide the mechanism for exploiting the outcome of each evaluation by archiving the (local or global) best-so-far individuals on the associated poles.

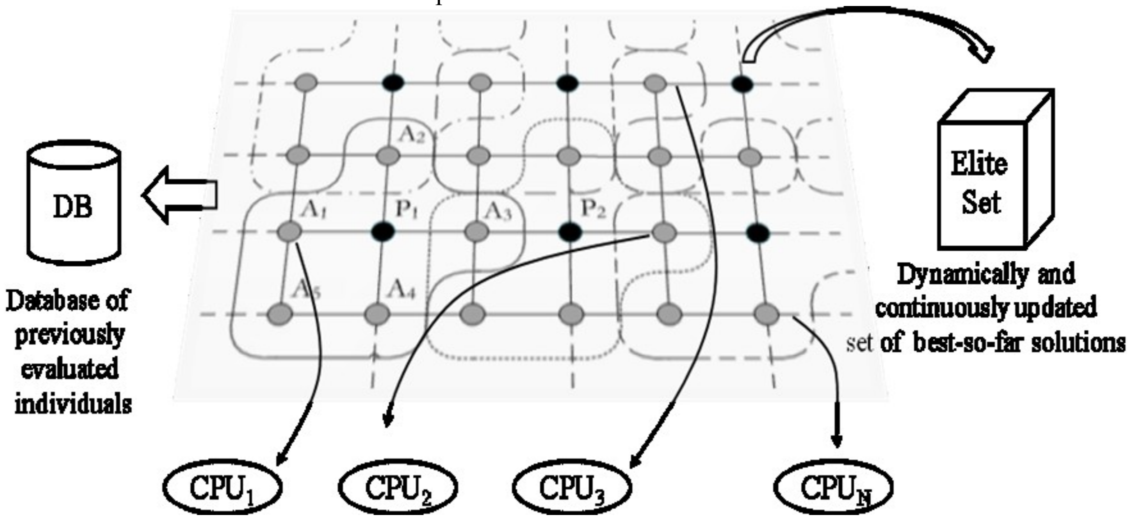
DB
Database of previously evaluated individuals
A2
A1
P1
A3
P2
A5
A4
Elite Set
Dynamically and continuously updated set of best-so-far solutions
CPU1
CPU2
CPU3
CPUN

Figure 3.Demes' topology on a 6×4 supporting mesh. Its six demes are clearly marked. Poles are marked with black circles. Regarding the deme associated with pole P_1, its agent A_3 is also shared with the adjacent deme associated with P_2 etc, whereas A_5 is the only non-shared agent. Dashed lines indicate the mesh periodicity.

During the initial phase of the algorithm evaluations are assigned to all available processors, by selecting agents and generating the corresponding individuals at random. Upon completion of any evaluation, the corresponding agent receives its objective function value(s) and a decision on whether the just evaluated individual must displace or not the affected pole(s) is to be made. This decision is based upon dominance criteria, by comparing the evaluated individual $\vec{x}_{agent}$ and the existing $\vec{x}_{pole}$. If $\vec{x}_{agent}$ dominates $\vec{x}_{pole}$, $\vec{x}_{pole}$ is displaced by $\vec{x}_{agent}$. If $\vec{x}_{pole}$ dominates $\vec{x}_{agent}$, no displacement takes place. In case that there is no absolute domination, other criteria including a partially stochastic selection are used. The non-dominated solutions form a dynamically updated elite set. Also, the values of all individuals paired with the corresponding objective function value(s) are stored in a database (DB).

The selection of the new individual to undergo evaluation on the CPU which just became idle is determined through a series of inter- and intra-deme processes, which are based on two priority metrics related to each agent's cost and age. More specifically, each pole is assigned a dynamically changing priority $\Pr_p$, which is the product of an age-based $\Pr_p^{age}$ and a cost-based $\Pr_p^{\cos t}$ priority, namely $\Pr_p = \Pr_p^{age} \Pr_p^{\cos t}$. For each pole, $\Pr_p^{age}$ is the average age of its agents; the age of an agent is defined as the difference of the serial number of the last evaluation carried out by this agent from that of the current evaluation. The cost-based priority $\Pr_p^{\cos t}$ is defined using strength- and density-based criteria applied to the individuals stored at the poles. The next agent to undergo evaluation is chosen from the deme of the pole with the highest priority $\Pr_p$. Within this deme, the agent with the maximum age is selected.

After having selected the agent, a new candidate solution is generated via an intra-deme process, through the application of recombination and mutation operators. The recombination scheme used is inspired by the mutant vectors used in differential evolution [7] and is restricted within the chosen deme. A non-uniform mutation scheme with a small user-defined probability is applied.

2.2 Implementation of Surrogate Evaluation Models

As in [3], without however the grid-enabling issues presented there, the AMAEA is based on an implementation of the inexact pre-evaluation (IPE) technique within the already described asynchronous algorithm. The idea of pre-evaluating individuals for indentifying whether each of them is worth a CFD-based re-evaluation can be found in previous publications by the same group, such as [4, 5, 6], which however are concerned with generation-based EAs. The IPE task is differently applied in asynchronous EAs (i.e. in AMAEAs, see also [3]) due to the inability to concurrently handle a population of candidate solutions. To perform the IPE task, the DB of previously evaluated individuals must be used. Once the number of DB entries exceeds a user–defined minimum, i.e. when the DB can provide enough patterns (hopefully adequately spread over the design space) for training the surrogate evaluation models, the IPE task starts. During the latter, instead of generating a single new individual to undergo evaluation per agent, N_{IPE} trial individuals are generated. All of them are associated with the same agent. The N_{IPE} value is user-defined. For each trial member, a local radial basis function (RBF) network [8] is trained using neighboring (with distances measured in the design space) individuals carefully selected from the DB. The training pattern selection algorithm can be found in [4]. For each trial individual, the RBF networks provide approximations to the objective function value(s). The later are rank sorted using a dominance metric and the "best" of them, according to the RBF network based predictions, is selected to undergo evaluation on the idle CPU. Note that selecting training patterns, training the RBF network and approximating the objective function value(s) have all together negligible CPU cost, compared to the cost of a single evaluation on the CFD software.

3 DESIGN-OPTIMIZATION OF THE HYDROMATRIX® RUNNER

The design-optimization of a matrix turbine runner using the aforementioned AMAEA is handled as a three objective optimization problem. All objectives must be minimized. Let us firstly define the objective functions for each operating point under consideration; these will de denoted by f_j, by temporarily omitting the superscript related to the operating point (see below). The first objective function (f_1) is related to the "smooth" interfacing of flows developed within the runner and the draft tube. Given that the draft tube is fixed, specific swirl and axial velocity distributions at the exit of the runner are required. Thus, f_1 is defined as the weighted sum of the deviations of the outlet swirl and axial velocity distributions from target curves defined by the designer. The second objective function (f_2) is related to the loading of the blades which must present the minimum variations over the blade surface; thus, f_2 expresses the standard deviation of the pressure distribution along the chordwise direction, at eleven equidistant spanwise locations. Finally, the third objective function (f_3) controls cavitation and equals to the cavitation index $\sigma = (p_{ref} - p_{min}) / \frac{1}{2}\rho U_{ref}^2$ [9] (p_{min} is the minimum pressure over the blade surface and p_{ref}, U_{ref} are the pressure and velocity at a reference point, respectively).

In this work, the runner is designed so as to perform optimally at the best efficiency point (BEP), one part- and one full-load points. They correspond to height H equal to 8.5, 4.2 and 10 meters, respectively. So, regarding the AMAEA, the three objectives (F_1, F_2 and F_3) to be minimized are the weighted sums of f_1, f_2 and f_3 at the three operating points,

$$F_i = \sum_{j=1}^{3} w_j f_i^{OP_j}$$

where i denotes the objective and j the operating point. In the above equation, w_j are appropriate weights. For the last two objectives, weights $w_1 = 1$ (at the BEP) and $w_2 = w_3 = 0.1$ (at the part- and full-load points) are employed. For the first objective $w_1 = 1$ and $w_2 = w_3 = 0$. Constraints on the mass flow rates at the three points are also imposed. For each of them, apart from the height H, the desirable mass flow rate is specified. Since the latter results from the CFD analysis, candidate geometries which cannot meet the value defined by the designer (practically, those that deviate more than $\pm 10\%$ from the target value) are considered as infeasible.

Recall that, this paper is dealing with the design of the runner blade only. However, each CFD-based evaluation is carried out over a flow domain which includes also the stator. The mixing plane model is used to cope with the stator-rotor interaction. The hub and shroud generatrices are given and the stator blades are fixed. The runner has 4 blades. Each runner blade shape is modeled by superimposing a known thickness distribution onto a parameterized mean camber surface (Figure 5). Since the blade thickness distribution cannot vary, the runner blade shape is controlled only by the parameters defining its mean camber surface. The design variables are the coordinates of the control points of the Bezier curves used to parameterize the spanwise distributions of (a) the mean camber surface angles at the leading (LE) and trailing (TE) edges, (b) the circumferential position of the blade LE and TE and (c) the mean camber surface curvature. Overall, there are 52 design variables.

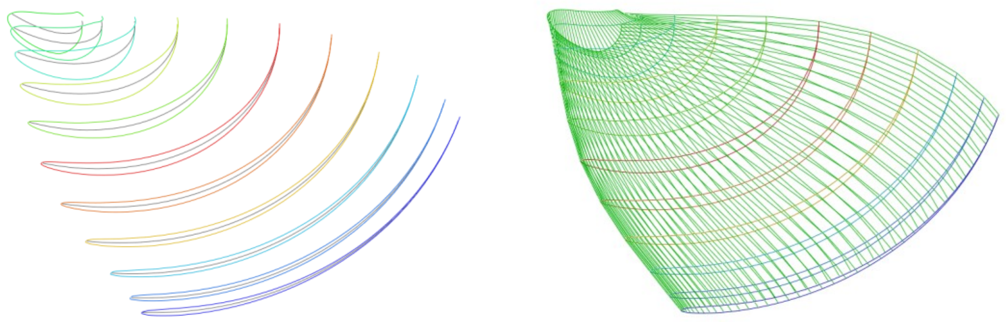

Figure 4. Schematic representation of the runner blade parameterization. Eleven spanwise blade profiles generated over the mean camber surface using the thickness distributions (left) and the surface grid of the blade (right) are shown.

During the evolutionary optimization, the evaluation of each candidate solution (i.e. each new runner geometry) consists on (i) the blade surface generation based on the corresponding values of the design variables (ii) the grid generation and (iii) the numerical solution of the flow (Euler) equations using software developed by Andritz-Hydro. These steps are schematically shown in Figure 5.

The CPU cost of each evaluation, including the numerical solution of the flow at all three operating points, may vary between 10 sec and 14 min on an Intel Core Duo processor. The difference in the CPU cost is due to the extended design space of the problem that may often lead to infeasible geometries for which the optimization algorithm skips their evaluation on the CFD software by just penalizing them. Furthermore, in the beginning of the optimization, a lot of individuals usually fail to satisfy the imposed mass flow constraints; these are also penalized by overcoming CFD-based evaluations. The existence of many infeasible solutions (either due to geometrical or flow constraints) is a good reason for selecting the asynchronous algorithm that enables the exploitation of all available computational resources without idle periods of time. Practically, each time an evaluation fails, a new individual to undergo evaluation is immediately assigned to the temporarily idle CPU. Note that, if a synchronous algorithm was used, some CPUs would often remain idle for a while, due to the synchronization barrier at the end of each generation.

This design was performed on 16 interconnected CPUs using an AMAEA with a 10×10 supporting mesh (i.e. with 75 agents and 25 poles). Metamodels were applied after the first 300 entries have been stored in the DB, by generating and approximately evaluating $N_{IPE}=8$ different trial members. The optimization was configured to terminate after 2000 CFD-based evaluations.

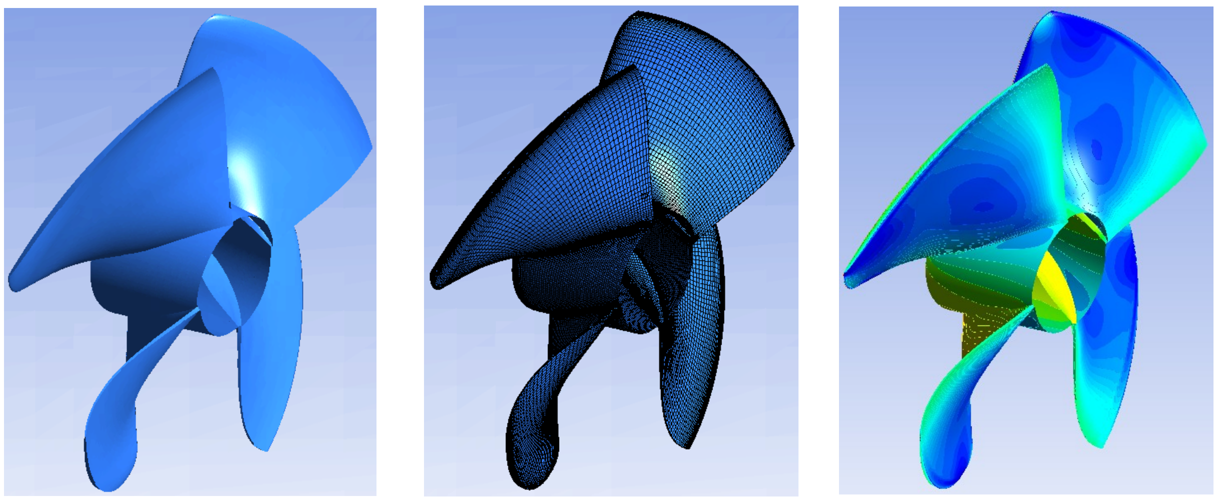

Figure 5. The three steps of the CFD-based evaluation of a candidate runner geometry. From left to right: blade surface generation, grid generation and the pressure field computed by the numerically solving the flow equations at a single operating point.

Figure 6 presents the front of non-dominated solutions computed by the AMAEA. For this front, the solutions stored at each of the 25 poles of the supporting mesh are shown in Figure 7. From the 24 non-dominated solutions of the computed front, 14 are identical to the solutions currently stored at the poles of the supporting mesh. This was expected since at the poles the best-so-far (non-dominated) solution of their deme is stored. In Figure 8, the evaluations performed by each agent (left part) and deme (right part), expressed as percentage (%) of the total number of evaluations, are also shown. One may notice that the evaluations are "equally" shared among the 25 demes; also, as expected, agents belonging exclusively to one deme (for instance agents 5, 7, 9, 25, etc) usually perform less evaluations than agents shared between two demes each.

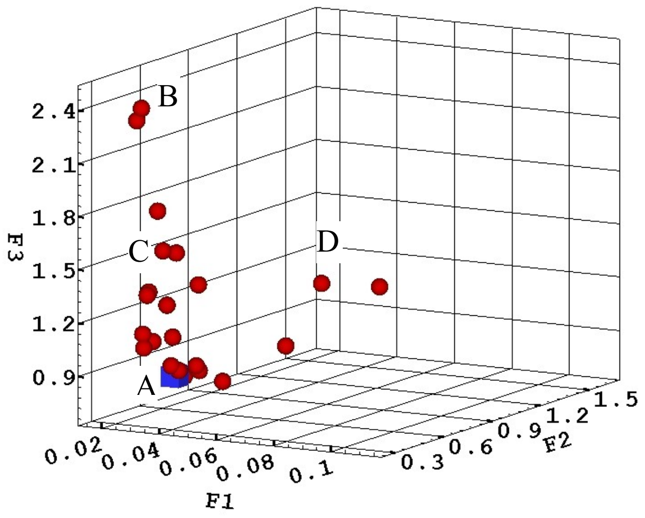

Figure 6. Front of non-dominated solutions computed at the expense of 2000 CFD-based evaluations.

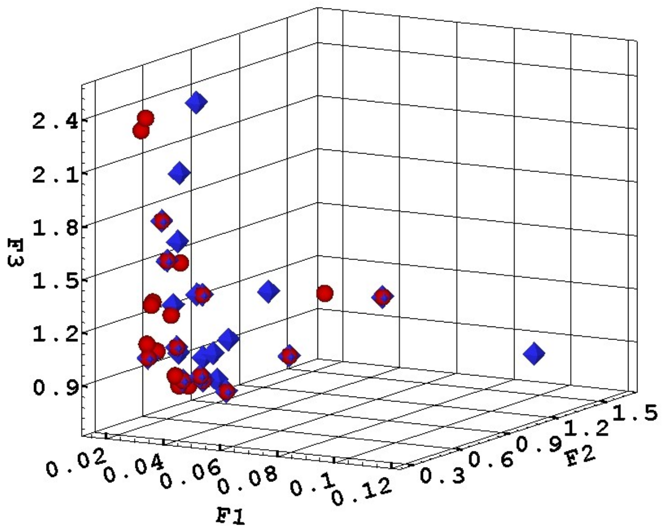

Figure 7. Front of the computed non-dominated solutions (red spheres), plotted along with the "local-best" solutions (blue octahedra) stored at the poles at the end of 2000 CFD-based evaluations.

An indicative solution (cube marked with A, Figure 6) selected from the computed front is further analyzed in detail and the obtained results are shown in Figures 9 to 12. In particular, Figure 9 presents the result of the numerical solution of the flow equations at the BEP point. The pressure coefficient distribution shown in Figure 10 confirms the very low cavitation danger at this operating point. The computed outlet velocity distributions (Figure 10, right) fit perfectly to the draft tube that follows. Concerning the full-load operating point (Figure 11), the pressure coefficient distributions indicate a small possibility of cavitation close to the shroud. The outlet axial velocity distribution remains reasonably good with a higher swirl. At the part-load operating point (Figure 12), the possibility of cavitation close to the shroud remains but this is moved towards the pressure side (see the crossing pressure profiles there). The outlet velocity distributions indicate an increase in swirl but in the opposite direction than at the BEP and full-load points. From the presented results, one may realize the changes in the performance characteristics at the three operating points. Recall that the weights at the BEP are an order of magnitude higher than those associated with the other two operating points. An immediate consequence is that, when operating at the part- and full-load points, with a predefined mass-flow, the possibility of cavitation is higher and, also, a worse matching with the draft tube is expected.

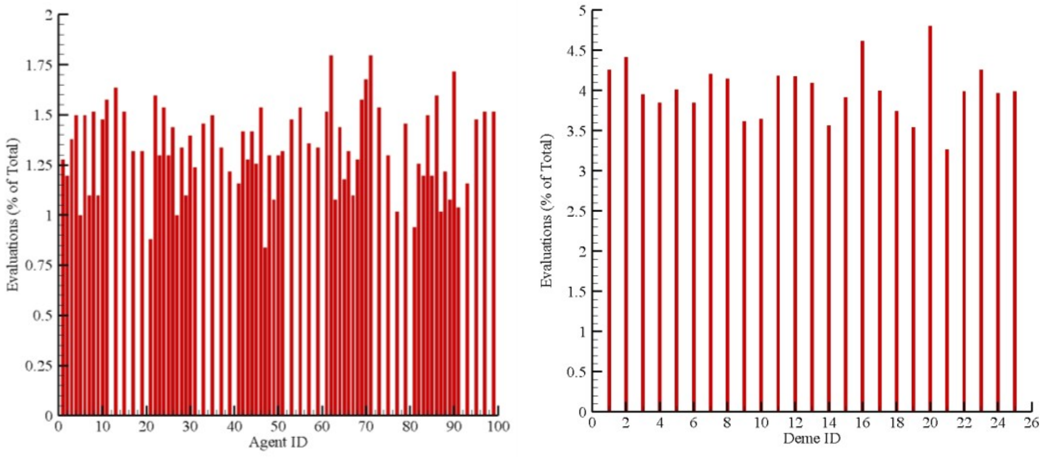

Figure 8. Evaluations performed by each of the supporting mesh agents (left) and demes (right), as a percentage (%) of the total number of evaluations.

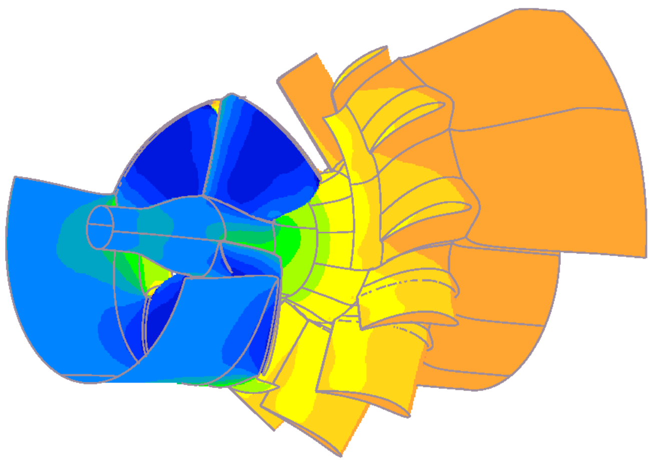
Figure 9. 3D view of the matrix turbine for the selected non-dominated solution marked with A in Figure 6. The predicted pressure field at the BEP is shown.

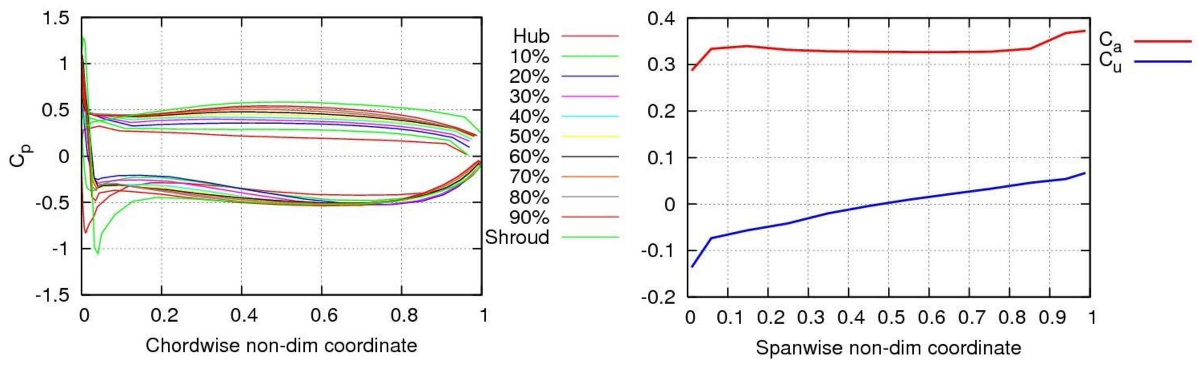

Figure 10. Detailed analysis of the non-dominated solution A, at the BEP. Chordwise pressure coefficient distributions at 11 equidistant spanwise locations (left) and normalized outlet velocity distributions (C_a: axial, C_u: peripheral; right). The non-dimensional spanwise coordinate measures from the hub to the shroud.

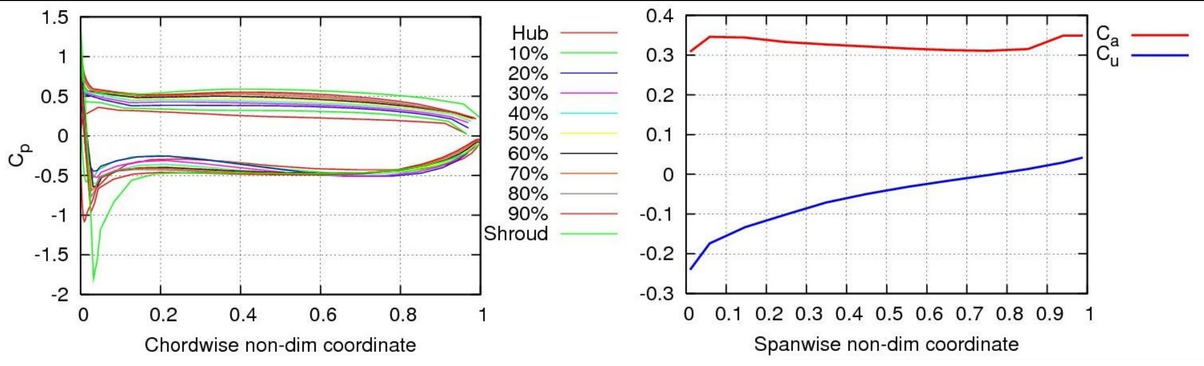

Figure 11. Detailed analysis of the non-dominated solution A, at the full-load point. Chordwise pressure coefficient distributions at 11 equidistant spanwise locations (left) and normalized outlet axial/peripheral velocity distributions (right).

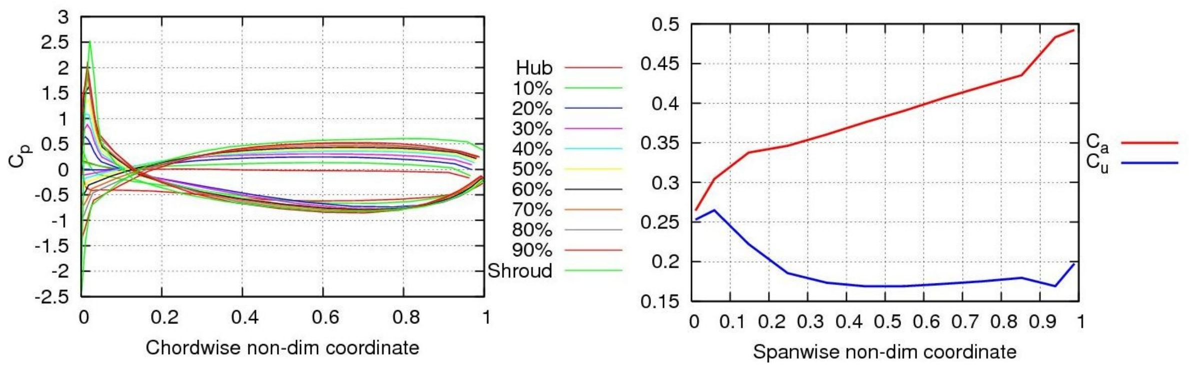

Figure 12. Detailed analysis of the non-dominated solution A, at the part-load operating point. Chordwise pressure coefficient distributions at 11 equidistant spanwise locations (left) and normalized outlet axial/peripheral velocity distributions (right).

To better understand the computed Pareto front characteristics, three more points (B, C and D; see Figure 6) selected from the front of non-dominated solutions are compared. Values of the objective functions (F_1, F_2, F_3), and their constituents (f_1, f_2, f_3), for each operating point, are shown in Table 1. Recall that F_1=f_1, so the optimization takes into account the quality of the velocity profiles at the runner exit, at the BEP only. Based on this table, solution D performs better with respect to f_1 at the BEP, ensuring optimal matching with the draft tube. Solutions A and B have equally low f_1 values (slightly greater than that of D) so that their matching with the existing draft tube is expected to be reasonably good. Based on Figure 13 (right) solution A, B and D give an almost flat axial velocity profile along the spanwise direction, being too close to the target distribution (not shown here). In contrast, for solution C, the axial velocity distribution in the spanwise direction varies a lot; this deviation from the flat target distribution is one of the reasons for the high f_1 value, see Table 1. Regarding, the peripheral velocity distributions, also shown in Figure 13 (right), solutions A, B and D are also very close to the target distribution (included in the figure). Solution C deviates from the target distribution and this also contributes to the increase of the f_1 (or F_1) value. At the BEP, solution C has almost constant loading along the blade surface (lower f_2 value) and this can be confirmed by Figure 13 (left).

Front Point	F_1	F_2	F_3	Best efficiency point			Full-load point		Part-load point	
				f_1	f_2	f_3	f_2	f_3	f_2	f_3
A	0.025	0.589	0.818	0.025	0.232	0.584	0.216	1.009	3.345	1.324
B	0.026	0.359	2.396	0.026	0.268	2.043	0.345	2.333	0.565	1.191
C	0.109	0.344	1.533	0.109	0.214	1.290	0.284	1.50	1.011	0.926
D	0.019	1.578	1.037	0.019	0.419	0.715	0.279	0.649	11.306	2.570

Table 1 : Objective functions values (F_1, F_2, F_3) along with their constituents (f_1, f_2, f_3), for the four non-dominated from the Pareto front.

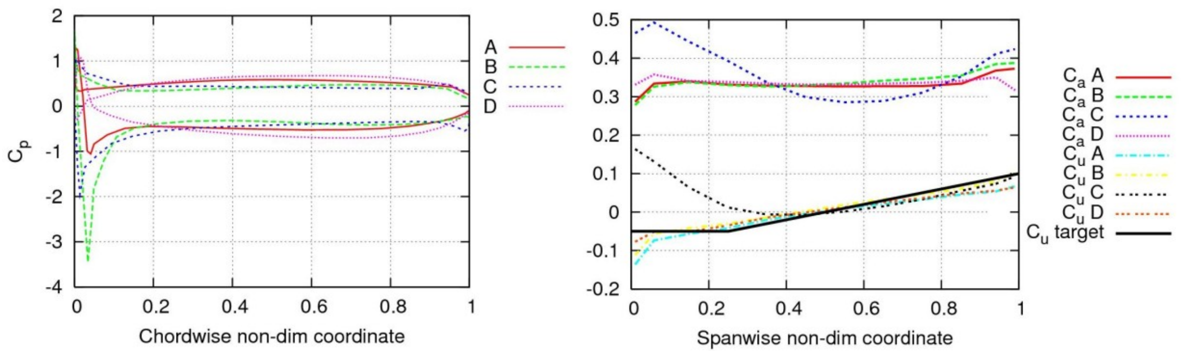

Figure 13. Comparison of the four selected non-dominated solutions at the BEP. Chordwise pressure coefficient distributions at the shroud (left) and normalized outlet velocity distributions (right).

At the full-load point, solution A has the best loading along the blade surface (lower f_2 value) and this is confirmed by Figure 14 (left). Regarding the cavitation index (f_3 value), solution D has the lowest cavitation risk at the full-load point. At the part-load point solution B, has the better blade loading (f_2) (see also Figure 14; right) whereas solution C is the safest with respect to cavitation.

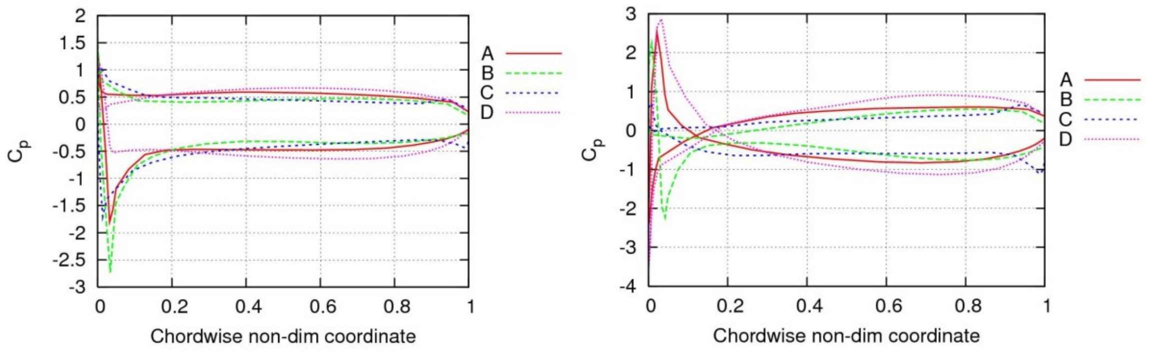

Figure 14. Chordwise pressure coefficient at the shroud of the four selected non-dominated solutions at the full- (left) and part-load points (right).

4 CONCLUSIONS

In this paper the design - shape optimization of a Hydromatrix® turbine runner, using an asynchronous metamodel-assisted evolutionary algorithm (AMAEA) was presented. Given that the optimization problem was highly constrained and the cost per evaluation may vary a lot, the use of the asynchronous algorithm presents noticeable advantages since this fully exploits the available CPUs by almost vanishing idle computing times. The design was carried out at three operating points, using three objectives per operating point and appropriate weights defined by the designer. The so-computed Pareto front of non-dominated solutions allows various decisions to be made afterwards, based on additional criteria.

AKNOWLEDGMENT

This work was funded by the EU project "*HYDROACTION – Development and laboratory testing of improved action and Matrix hydro turbines designed by advanced analysis and optimization tools*" (PROJECT NUMBER 211983).

REFERENCES

[1] www.hydromatrix.at

[2] Asouti, V.G. and Giannakoglou, K.C. (2009),"Aerodynamic optimization using a parallel asynchronous evolutionary algorithm controlled by strongly interacting demes", *Engineering Optimization*, Vol. 41, pp. 241-257.

[3] Asouti, V.G., Kampolis, I.C. and Giannakoglou, K.C. (2009),"A Grid-enabled asynchronous metamodel-assisted evolutionary algorithm for aerodynamic optimization", *Genetic Programming and Evolvable Machines (SI: Parallel and Distributed Evolutionary Algorithms, Part One)*, Vol. 10, pp. 373-389.

[4] Karakasis, M.K. and Giannakoglou, K.C. (2006),"On the use of metamodel-assisted, multi-objective evolutionary algorithms", *Engineering Optimization*, Vol. 38, pp. 941-957.

[5] Giannakoglou, K.C. (2002),"Design of Optimal Aerodynamic Shapes using Stochastic Optimization Methods and Computational Intelligence", *Progress in Aerospace Sciences*, Vol. 38, pp. 43-76.

[6] Giannakoglou, K.C, Giotis, A.P. and Karakasis, M.K. (2001),"Low-cost genetic optimization based on inexact pre-evaluations and the sensitivity analysis of design parameters", *Inverse Problems in Engineering,* Vol. 9, pp. 389-412.

[7] Storn, R., Price, K. (1997)," Differential Evolution - A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, Vol. 1, pp. 341-359.

[8] Haykin, S. (1999), *Neural networks: A comprehensive foundation,* Prentice Hall, New Jersey, USA.

[9] Krivchenko, G. (1994), *Hydraulic machines: turbines and pumps,* Lewis Publishers, CRC Press Inc.