Bohumír Bastl; Marek Brandner; Jiří Egermaier; Hana Horníková; Kristýna Michálková; Eva Turnerová
Gradient-free and gradient-based methods for shape optimization of water turbine blade

In: Jan Chleboun and Pavel Kůs and Petr Přikryl and Miroslav Rozložník and Karel Segeth and Jakub Šístek and Tomáš Vejchodský (eds.): Programs and Algorithms of Numerical Mathematics, Proceedings of Seminar. Hejnice, June 24-29, 2018. Institute of Mathematics CAS, Prague, 2019. pp. 15–26.

Persistent URL: http://dml.cz/dmlcz/703072

Programs and Algorithms of Numerical Mathematics 19
J. Chleboun, P. Kůs, P. Přikryl, M. Rozložník, K. Segeth, J. Šístek, T. Vejchodský (Eds.)
Institute of Mathematics CAS, Prague 2019

GRADIENT-FREE AND GRADIENT-BASED METHODS FOR SHAPE OPTIMIZATION OF WATER TURBINE BLADE

Bohumír Bastl, Marek Brandner, Jiří Egermaier,
Hana Horníková, Kristýna Michálková, Eva Turnerová

University of West Bohemia in Pilsen, Faculty of Applied Sciences
Univerzitní 22, 306 14, Plzeň, Czech Republic
bastl@kma.zcu.cz, brandner@kma.zcu.cz, jirieggy@kma.zcu.cz, hhornik@kma.zcu.cz,
krmich@ntis.zcu.cz, turnerov@kma.zcu.cz

Abstract: The purpose of our work is to develop an automatic shape optimization tool for runner wheel blades in reaction water turbines, especially in Kaplan turbines. The fluid flow is simulated using an in-house incompressible turbulent flow solver based on recently introduced isogeometric analysis (see e.g. J. A. Cotrell et al.: Isogeometric Analysis: Toward Integration of CAD and FEA, Wiley, 2009). The proposed automatic shape optimization approach is based on a so-called hybrid optimization which combines gradient-free and gradient-based methods. As the gradient-free method, the Particle Swarm Optimization (PSO) method is used. The gradient-based part exploits a quasi-Newton method implemented in IpOpt software library (Interior Point OPTimizer) and gradients of the objective function with respect to design variables are provided by automatic differentiation of the computer code which is done with the help of CoDiPack software library (Code Differentiation Package).

Keywords: shape optimization, automatic differentiation

MSC: 65N15, 65M15, 65F08

1. Introduction

Automatic shape optimization is a very complex problem itself, even for simple test cases. Combined with modeling of real-world objects, like water turbines in our case, it is an extremely challenging task. This is the reason why these methods are frequently studied nowadays in the context of real-world applications.

In our case, the main aim of the automatic shape optimization is to improve the turbine efficiency in a wide range of operating conditions, i.e., to optimize the runner blade shape in such a way that the function describing efficiency with respect to the unit flow rate is as "high and flat" as possible. The flow in water turbines is

DOI: 10.21136/panm.2018.02

described by the motion of an incompressible Newtonian viscous fluid, i.e., by the Navier–Stokes equations, or the RANS (Reynolds-Averaged Navier-Stokes) equations with a suitable turbulent model (the reader is referred to [1], [2] for more details). In general, the optimization problem can be stated as

$$\min_{\boldsymbol{\phi}} J(\mathbf{u}(\boldsymbol{\phi}), p(\boldsymbol{\phi}), \Omega(\boldsymbol{\phi})) \tag{1}$$

subject to

$$R(\mathbf{u}, p, \Omega) = 0 \quad \text{in} \quad \Omega \subset \mathbb{R}^n, \tag{2}$$

where J denotes the objective function, which is specified later in this paper. The velocity $\mathbf{u}$ and the pressure p are obtained as the solution of the Navier–Stokes/RANS equations (2) on the domain Ω. In the optimization process, the shape of the runner blade, which is given by selected design parameters $\boldsymbol{\phi}$, changes. Thus, the computational domain Ω and also the velocity $\mathbf{u}$ and the pressure p depend on the design variables $\boldsymbol{\phi}$.

In general, there are two main approaches to shape optimization problems – gradient-free or gradient-based methods. Gradient-free methods are heuristics used for searching the whole design space and getting close to the global minimum of the given objective function, but suffer from huge computational demands, especially in cases with a large number of optimization parameters. On the other hand, gradient-based methods can be less computationally demanding because they are, more or less, independent of the number of optimization parameters (if a suitable method for gradient computation is used). Further, they provide fast convergence to the nearest local minimum. However, the obvious disadvantage is that gradients of the objective function with respect to the optimization parameters are necessary. It can be very difficult to compute them, or it is at least computationally expensive. The so-called hybrid methods combine gradient-free and gradient-based methods to get the best from both. We chose to adopt this approach in the proposed automatic shape optimization tool described in this paper.

2. Automatic shape optimization

Our main goal is to develop a tool that automatically performs shape optimization of a runner blade of a water turbine, with a special focus on Kaplan turbines, such that the turbine efficiency is improved in a wide range of operating conditions. In this section, we will give a brief overview of the proposed method.

Our approach is based on a hybrid optimization method that combines a gradient-free and a gradient-based method. For the gradient-free part, the Particle Swarm Optimization (PSO) method is used (see e.g. [8]), which performs well in many situations as it is reported in many papers. This part especially serves for searching the global minimum in the design space. For the gradient-based part, a combination of two software packages – IpOpt (see [5]) and CoDiPack (see [3]) – is used. CoDiPack is a package for gradient evaluation in computer codes and is based on automatic

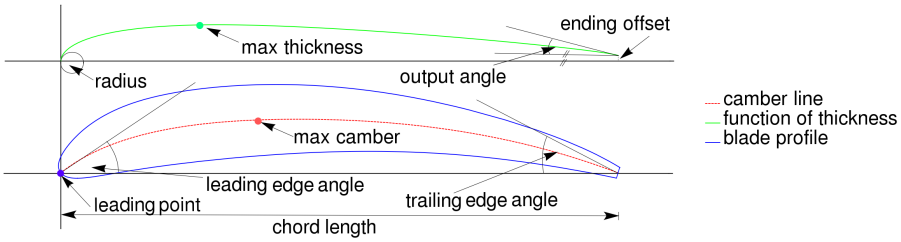

Figure 1: Design parameters determining a planar blade profile

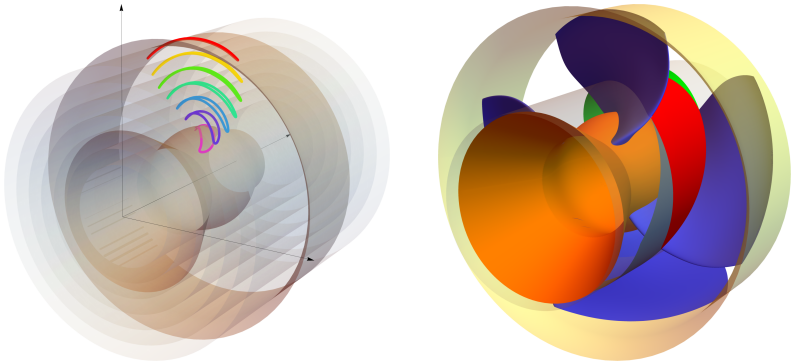

Figure 2: Left: Spatial blade profiles on cylinders. Right: The runner blades (blue) in a runner wheel domain bounded by inner (orange) and outer (yellow) surfaces.

differentiation. Thus, this package is responsible for providing gradients of the objective function with respect to the design variables. The gradient-based method itself is driven by IpOpt and a quasi-Newton method implemented there. The main aim of this part is to obtain better approximation of the nearest minimum, i.e., improve the solution found by the PSO method.

Before describing the objective function, it is necessary to mention some details about geometric modeling of water turbines that are needed for understanding the text below. The most important part is the modeling of the 3D shape of the runner blade. The shape of the runner blade is typically determined by 7–11 planar blade profiles. The shape of each planar blade profile is obtained by composing a so-called camber line with a thickness function, which are determined by several parameters (see Fig. 1). These parameters subsequently serve as the design parameters in the optimization process. In our case, the camber line and the thickness function are described by B-spline objects given by the design parameters, and subsequently also the B-spline description of the planar blade profile is constructed. Further, each planar blade profile is mapped to a cylinder of a given radius with axis coincident with the turbine axis. Then, the B-spline description of the corresponding spatial blade profiles is constructed (see Fig. 2 (left)). The final shape of the runner blade is obtained as a loft B-spline surface interpolating these spatial blade profiles (see [9] for more details on loft B-spline surfaces), see Fig. 2 (right).

An important part of any optimization is the choice of the objective function. For computational reasons, our optimization method is based on the optimization of the planar blade profiles determining the shape of the runner blade. This leads to the following choice of the objective function

$$J = w_1(-J_1) + w_2 J_2 + w_3 J_3, \tag{3}$$

which is a weighted combination of three terms (w_1, w_2, w_3 represent the weights of these terms, typically taken from $[0, 1]$ and fulfilling $w_1 + w_2 + w_3 = 1$):

- Total lift of the planar blade profile

$$J_1 = \int_{b_p} p\boldsymbol{n} \cdot \boldsymbol{l} \mathrm{d}s,$$

  where b_p is the curve representing the blade profile, $\boldsymbol{n}$ is the unit outer normal to the blade profile and $\boldsymbol{l}$ is the unit direction opposite to the blade motion. This term is related to the turbine efficiency because by increasing the value of J_1, more energy is taken from the water by the runner blade. Negative sign of J_1 in (3) reflects the fact that the lift of the profile is maximized, whereas the objective function J is minimized.

- Velocity term

$$J_2 = \int_{b_{\text{out}}} (u_t - u_{\text{targ}})^2 \mathrm{d}s,$$

  where u_{targ} is the target value of the tangential velocity and b_{out} is the outflow boundary of the computational domain. This term reflects the fact that the interaction of the runner wheel and the draft tube behind the runner wheel is essential for the overall efficiency of the turbine. The value of u_{targ} is based on the experience from practice and preserves the optimal flow in the draft tube.

- Pressure term

$$J_3 = \int_{b_{p,\text{press}}} (p - p_{\text{targ, press}})^2 \mathrm{d}s + \int_{b_{p,\text{suc}}} (p - p_{\text{targ, suc}})^2 \mathrm{d}s,$$

  where $p_{\text{targ, press}}$ and $p_{\text{targ, suc}}$ are targets value of pressure at pressure and suction side of the blade profile, respectively. The idea of this term is to force the pressure distribution along the pressure and suction side of the blade profile to be as even as possible, which means that also the energy of water is evenly taken along the blade profile, without any strokes. Thus, it also helps to prevent cavitation. The specific values of $p_{\text{targ, press}}$ and $p_{\text{targ, suc}}$ for each blade profile are derived from the pressure distribution along the initial blade profile (from which the optimization process starts).

Instead of the values of the objective function, it is also possible to monitor their relative change with respect to the values of the objective function terms of the initial design, i.e.,

$$J_{1,\mathrm{rel}} = \frac{L_{\mathrm{init}}}{J_1}, \quad J_{2,\mathrm{rel}} = \frac{J_2}{V_{\mathrm{init}}}, \quad J_{3,\mathrm{rel}} = \frac{J_3}{P_{\mathrm{init}}},$$

where L_{init}, V_{init}, P_{init} correspond to J_1, J_2, J_3 evaluated for the initial planar blade profile, respectively. Of course, the initial values of the components of the objective function differ for each planar blade profile. Then the objective function is defined as

$$J = w_1 J_{1,\mathrm{rel}} + w_2 J_{2,\mathrm{rel}} + w_3 J_{3,\mathrm{rel}}. \tag{4}$$

Before describing the algorithm of the automatic shape optimization tool, we will briefly review the fundamentals of the PSO method. A swarm (population) in the i-th generation consists of N particles (members) such that each particle has its position $\mathbf{x}_j^i$ and velocity $\mathbf{v}_j^i$, $j = 1, \ldots, N$. The movement of each particle is influenced by its best known local position $\mathbf{p}_j^i$, $j = 1, \ldots, N$, and is also guided toward the particle of the population $\mathbf{g}^i$ with the best global position in the search space. To obtain the swarm in the $(i+1)$-th generation, first the velocities of the particles are updated via

$$\mathbf{v}_j^{i+1} = K(\mathbf{v}_j^i + c_1 r_1(\mathbf{p}_j^i - \mathbf{x}_j^i) + c_2 r_2(\mathbf{g}^i - \mathbf{x}_j^i)), \quad j = 1, \ldots, N,$$

where

$$K = \frac{2}{|2 - \Psi - \sqrt{\Psi^2 - 4\Psi}|}, \quad \Psi = c_1 + c_2$$

and r_1, r_2, c_1 and c_2 are random numbers. Then, the particle positions are updated via

$$\mathbf{x}_j^{i+1} = \mathbf{x}_j^i + \mathbf{v}_j^{i+1}, \quad j = 1, \ldots, N.$$

Finally, the objective function is evaluated for each new particle of the swarm and the best positions $\mathbf{p}_j^{i+1}$, $j = 1, \ldots, N$, and $\mathbf{g}^{i+1}$ are updated.

Now, we will give an overview of the proposed automatic shape optimization tool and we will comment on some steps of this approach in more details after this summarization.

1. For given operational conditions, like head or required flow rate, find the initial design of the runner blade which is given by design parameters for a selected number of planar blade profiles.

2. Precompute the initial values of all terms of the objective function (3) for all planar blade profiles.

3. Choose the weights w_1, w_2, w_3 (see (4)) for the follow-up optimization process.

4. Choose the parameters of the follow-up optimization process:

- the size of the population in the swarm method;
- the number of cycles of the swarm method;
- the number of generations in each cycle of the swarm method;
- whether you want to use the gradient-based optimization or not (if so, choose also its parameters, e.g. the maximum number of iterations of this gradient-based optimization method).

5. For each planar blade profile do:
   (a) For each cycle of the swarm method do:
      i. Generate the population of the swarm.
      ii. For each generation of the population in the cycle do:
         A. For each member of the population do:
            - generate the B-spline computational mesh;
            - compute the fluid flow simulation;
            - evaluate the objective function (see (4)).
         B. Modify the members (i.e., design parameters) of the population.
   (b) Use the gradient-based optimization to further improve the best member found during the last cycle of the swarm method.

Step 5(a) is related to the fact that the velocity of the members of the population typically quickly decreases in later generations of the PSO method. Thus, it does not make much sense to run the optimization process in one cycle with many generations because the swarm will get stuck around some point and becomes "stiff". It is more efficient to restart the swarm after several generations starting with the best member found to this point, i.e., take the best member found so far and generate again the population randomly around this best member.

Step 5A contains one of the most computationally intensive parts of the algorithm because it involves the turbulent fluid flow simulations for each member of the population. The first necessary part is to generate the geometry of the computational domain, i.e., generate the B-spline mesh between the two neighboring planar blade profiles in the blade net. Note that the generated B-spline mesh has to be refined (especially near the blade profile) as the viscosity of the simulated fluid is low. The example of such B-spline mesh is shown in Fig. 3 (left). This B-spline mesh is a result of a long-term searching of a compromise between the quality of the results and the number of degrees of freedom. Fig. 3 (right) then shows an example of the resulting pressure distribution obtained by the turbulent flow solver based on LRN $k-\omega$ model (see [6]). Currently, we use a setting of initial and boundary conditions for the turbulent model which is again a result of a long-term testing for this particular problem, as no general setting, which works universally, exists. We compute an unsteady flow simulation with a stopping criterion reflecting the relative

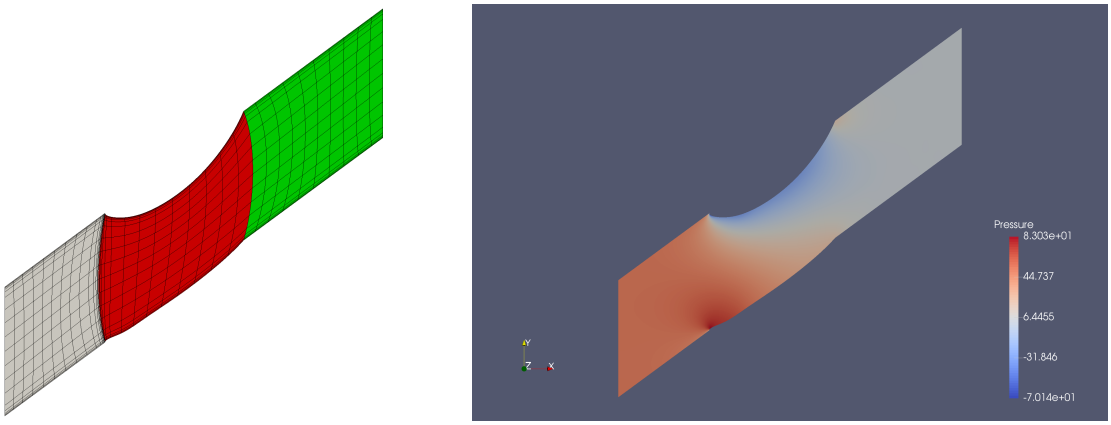

Figure 3: Left: Multi-patch computational domain with the B-spline mesh for one of the blade profiles (the closest one to the inner surface of the water turbine). Right: Pressure computed by the turbulent flow solver.

change of the solution between two consequent time iterations and/or the relative change of the objective function (4). With our setting, the given stopping criterion is typically met in less than 200 time steps, but in tens of time steps later in the optimization process, since we use the solution from the previous generation of the swarm method as the initial one for the next generation.

Step 5(b) covers running the gradient-based optimization which starts from the best member of the population found during the last cycle of the swarm method and this step is optional. CoDiPack is used to differentiate the code performing the Step 5A in order to evaluate the gradients of the objective function with respect to the design parameters.

3. Example

In this section, we will demonstrate the functionality of the proposed shape optimization algorithm on a particular example. In this example, the following basic parameters of the Kaplan turbine are given:

- head $H = 10\ m$;
- flow rate $Q = 5.73\ m^3/s$;
- unit speed $n_{11} = 170\ min^{-1}$.

In the following, we make some comments and mention some details related to selected parts of the proposed optimization process related to this example.

In this case, the shape of the runner blade is determined by 7 planar blade profiles. Thus, the optimization problem is solved for each of these 7 planar blade profiles and each of them is determined by 10 design parameters. Assuming that the chord length

and the ending offset are not taken as design parameters, we have 8 design parameters left for the optimization: maximal camber v, position of maximal camber d, leading edge angle β_1, trailing edge angle β_2, maximal thickness v_t, position of maximal thickness d_t, output angle γ, radius r_o of an osculating circle at the leading point.

In this example, the kinematic viscosity $\nu = 10^{-5}$, the time step $\Delta t = 10^{-3}$ and the turbulent intensity $I = 5\%$ are set. The computational domain between the two blade profiles is composed of three patches, each of which is represented as a B-spline surface (see Fig. 3 (left)), and provides 7085 degrees of freedom. This domain represents the domain between two runner blades lying on the cylinder corresponding to the given blade profile and is obtained by unfolding this cylinder (see Fig. 2 (right), where the domain composed of gray, red and green subdomains corresponds to the domain depicted in Fig. 3 (left)). We set the following boundary conditions:

- inflow boundary (the left side of the left patch)
  - For the RANS equations, the inflow velocity is set by its components $\boldsymbol{u}_{\text{in}} = (u_{\text{in},m}, u_{\text{in},t})$, where $u_{\text{in},m}$ is obtained from the prescribed flow rate Q through the domain and $u_{\text{in},t}$ is computed such that the direction of $\boldsymbol{u}_{\text{in}}$ coincides with the direction of the tangent of the camber line of the profile at the leading point, i.e.,

    $$u_{\text{in},m} = \frac{4Q}{\pi(D^2 - d^2)}, \quad u_{\text{in},t} = u_{\text{in},m} \tan\alpha,$$

    where $Q = 5.73\ m^3/s$, $D = 1\ m$, $d = 0.245852\ m$ (in this particular case) and α is the angle between the tangent of the camber line at the leading point and the x-axis.
  - For the $k - \omega$ turbulent model, we set Dirichlet boundary conditions as follows

    $$k_{\text{in}} = \frac{3}{2}(UI)^2, \quad \omega_{\text{in}} = \frac{k_{\text{in}}}{\nu\nu_r},$$

    where $U = ||\boldsymbol{u}_{\text{in}}||$ and ν_r is the chosen viscosity ratio of the turbulent viscosity to the kinematic viscosity at the inflow boundary.
- outflow boundary (the right side of the right patch)
  - Homogeneous Neumann boundary conditions are used for the RANS equations and also for the $k - \omega$ turbulent model.
- blade profile (the top and the bottom curve of the middle patch)
  - For the RANS equations, the velocity is set to zero (solid wall), i.e., $\boldsymbol{u}_{\text{blade}} = \boldsymbol{0}$.

– For the $k-\omega$ turbulent model, we set

$$k_{\text{blade}} = 0, \quad \omega_{\text{blade}} = \frac{6\nu}{0.0708(y_1)^2},$$

where y_1 is the distance of the first grid point from the nearest wall.

- at the remaining boundaries, i.e., the top and bottom sides of the left patch and the right patch

  – Periodic boundary conditions are set to simulate the behaviour of the whole blade net.

We set the following parameters for the PSO method:

- the size of the population: 30;
- the number of PSO cycles: 4;
- the number of generations in each PSO cycle: 5.

It is important to note that we have also added several constraints on design parameters which limit the design parameters to reasonable ranges or prescribe reasonable relations between the design parameters:

- To prevent the occurrence of an inflection point on the camber line or the thickness function, i.e., to prevent an "S shape" of these curves, we require the following constraints to be fulfilled

$$\frac{v}{d} < \tan\beta_1, \quad \frac{v}{1-d} < \tan\beta_2, \quad \frac{v_t - k_t}{1-d_t} < \tan\gamma.$$

- To limit the size of the osculating circle at the leading point with respect to the thickness of the blade profile, we require $r_o < v_t/2$.
- The sum of the leading and trailing angle must not be too large, it must be less than $\pi/2$ for theoretical reasons, i.e., we require $\beta_1 + \beta_2 < \pi/2$.

We also set bounds for some of the design parameters which are based more on our intuition than on theoretical assumptions. For example, we use $0.25 < d < 0.75, 0.25 < d_t < 0.75$. For handling the constraints in the optimization method, we use the penalty function approach presented in [7].

In the experiments, we chose several different combinations of weights for the components of the objective function (4) to see how it affects the results. The choices of weights are presented in the table below

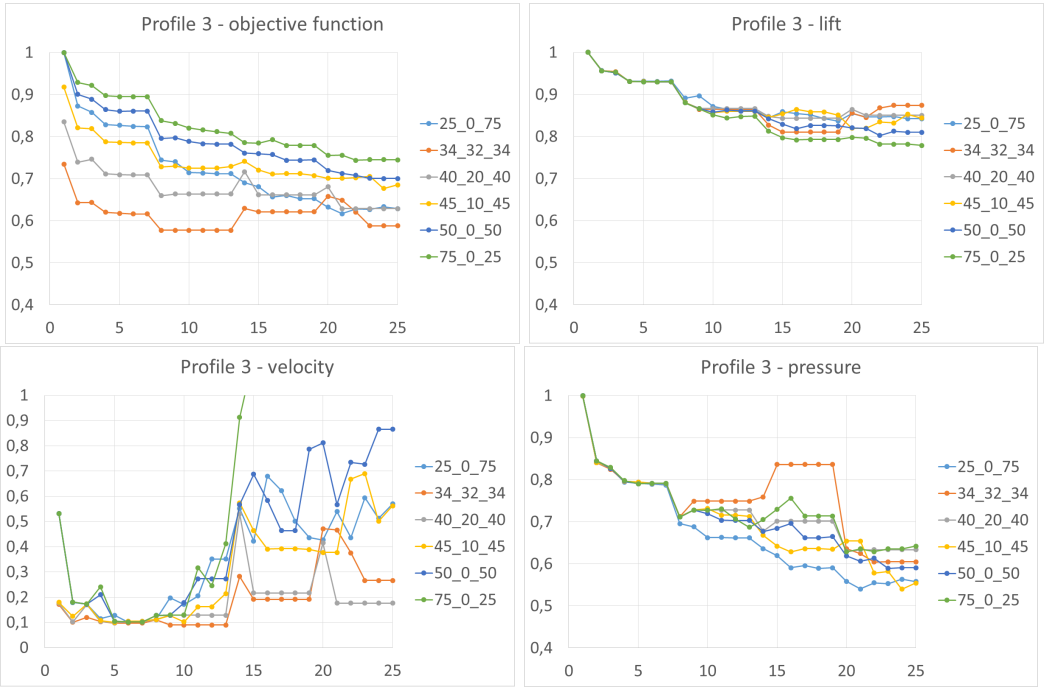

Figure 4: Evolution of the objective function and its parts for one of the blade profiles.

	w_1	w_2	w_3
1.	0.25	0	0.75
2.	0.34	0.32	0.34
3.	0.4	0.2	0.4
4.	0.45	0.1	0.45
5.	0.5	0	0.5
6.	0.75	0	0.25

Because of the space limitation, only some selected results will be presented. Fig. 4 shows the evolution of the objective function for all variants of the weights and also the evolution of all terms of the objective function during the iterations of the optimization process for one of the blade profiles. The results for the other blade profiles are similar. One of the observations that we can make is that the convergence of the optimization process is better for the choices of weights, where the weight for the velocity term is zero. On the other hand, for these choices of weights the velocity term tends to increase, especially for the blade profiles closer to the runner wheel hub. This does not have to be a problem because the initial blade profiles provide tangential velocity close to the target one and, thus, the increase even in multiples of the initial values does not have to mean that the blade is bad from this point of view. It can still provide tangential velocity close to the target value. Further, one can see an obvious drop in the objective function value after each restart of the

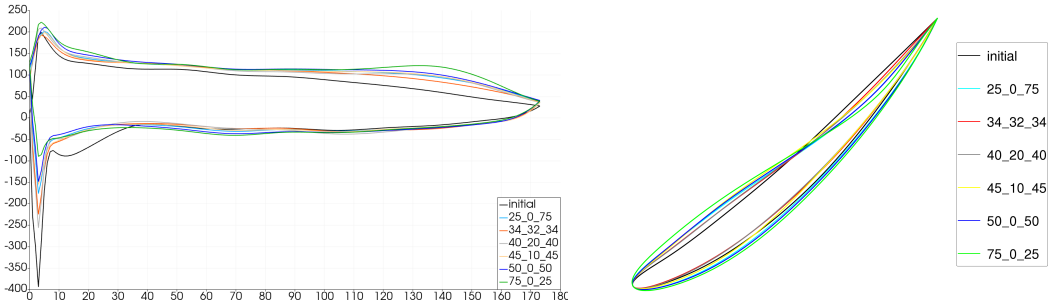

Figure 5: Left: Comparison of pressure distributions along the optimized and the initial planar blade profiles for one of the blade profiles. Right: Comparison of the final shapes provided by the optimization process for one of the blade profiles.

PSO method in most cases, whereas the optimization process tends to stagnate in between.

We also present the comparison of the pressure distributions for all variants of the weights together with the initial pressure distribution for one of the blade profiles in Fig. 5 (left). The most visible outcome here is an obvious trend to make the pressure distribution "flatter" on both sides of the blade profile. This is driven by the pressure term of the objective function, as we mentioned above. A comparison of the particular shapes of the resulting planar blade profiles is shown in Fig. 5 (right). For the presented case, we can see that the optimization process makes the blade profiles a little bit thicker and tends to move the position of the maximum camber more to the right.

Finally, we mention some computational and implementation details. As the PSO method requires analyzing a large number of swarm members in each generation, this step is parallelized to exploit the computational power of modern computers with many cores. Parallelization is also beneficial during gradient evaluation with the help of CoDiPack, where "forward-mode" is used, i.e., each derivative of the objective function with respect to some design variable is computed separately and in parallel. This approach is used because the other option, i.e., the use of "reverse-mode" of gradient evaluation, is extremely memory-demanding in this case. The computational time required for running optimization of one runner blade depends on specific settings of the method. For the example presented above, the optimization of the runner blade took about 2.5 days on a computer with 2×10 cores CPU Intel Xeon E5-2630 v4 @ 2.20GHz using 15 cores without the gradient-based optimization, and about twice as much time with the gradient-based optimization.

4. Conclusions

In this paper, an overview of a new automatic shape optimization tool for water turbines (especially, for Kaplan turbines) was presented. In our future work, we will focus on several improvements of the presented method, like generaliza-

tion to 2.5D case (combination of 2D turbulent flow simulations and 3D objective function evaluations) and later to 3D case (3D turbulent flow simulations and objective function evaluations). However, the full 3D case of the automatic optimization process is extremely computationally expensive at the moment and requires significant improvements of the in-house turbulent flow solver (faster solving of large linear systems, improvements in stabilization of the numerical solution and an adaptive refinement of the computational mesh).

Acknowledgements

The work leading to these results has received funding from the European Community's Horizon 2020 Programme (2014-2020) under grant agreement no. 678727.

References

[1] Bastl, B., Brandner, M., Horníková, H., and Šourek, J.: Report and prototype software for incompressible flow solver. Technical Report. University of West Bohemia in Pilsen, 2017, (available at `http://motor-project.eu/wp-content/uploads/2017/04/MOTOR_D3.2.pdf`).

[2] Bastl, B., Brandner, M., Horníková, H., and Turnerová, E.: Report: Incompressible turbulent flow solver. Technical Report. University of West Bohemia in Pilsen, 2018, (available at `http://motor-project.eu/wp-content/uploads/2018/02/MOTOR_D3.6.pdf`).

[3] CoDiPack `https://www.scicomp.uni-kl.de/codi/index.html` (accessed 14/08/2018).

[4] Cottrell, J. A., Hughes, T. J. R., and Bazilevs, Y.: Isogeometric analysis: toward integration of CAD and FEA. John Wiley & Sons, Ltd., 2009.

[5] IpOpt `https://www.coin-or.org/Ipopt/documentation/` (accessed 14/08/2018).

[6] Low Reynolds number version of Wilcox (2006) k-omega two-equation model `https://turbmodels.larc.nasa.gov/wilcox.html` (accessed 16/02/2017).

[7] Parsopoulos, K. and Vrahatis, M.: Particle swarm optimization method for constrained optimization problem. Front. Artif. Intell. Appl. **76** (2002), 214–220.

[8] Pehlivanoğlu, Y. V., Ay, S., and Gül, F.: Improved particle swarm optimization method directed by indirect surrogate modeling. J. Aeronautics Space Technologies **8** (2015), 1–10.

[9] Piegl, L. and Tiller, W.: The NURBS Book. Monographs in Visual Communication. Springer, Berlin Heidelberg, 1997.