EPJ Web of Conferences **114**, 02079 (2016)
DOI: 10.1051/epjconf/201611402079

Optimization of hydraulic turbine diffuser

Prokop Moravec[1, a], Juraj Hliník[2] and Pavel Rudolf[1]

[1]*Victor Kaplan Dept. of Fluid Engineering, Faculty of Mechanical Engineering, Brno University of Technology, Czech Republic*
[2]*Dept. of Mathematical Engineering, Faculty of Mechanical Engineering, Brno University of Technology, Czech Republic*

Abstract. Hydraulic turbine diffuser recovers pressure energy from residual kinetic energy on turbine runner outlet. Efficiency of this process is especially important for high specific speed turbines, where almost 50% of available head is utilized within diffuser. Magnitude of the coefficient of pressure recovery can be significantly influenced by designing its proper shape. Present paper focuses on mathematical shape optimization method coupled with CFD. First method is based on direct search Nelder-Mead algorithm, while the second method employs adjoint solver and morphing. Results obtained with both methods are discussed and their advantages/disadvantages summarized.

1 Introduction

Swirl turbine [1] is suited for low heads and relatively large discharges. Almost a half of the available head should be exploited in the draft tube by recovery of kinetic energy into pressure energy. Velocity field in the draft tube is characterized by swirl component on its inlet. This imposes stress on the draft tube performance [2].

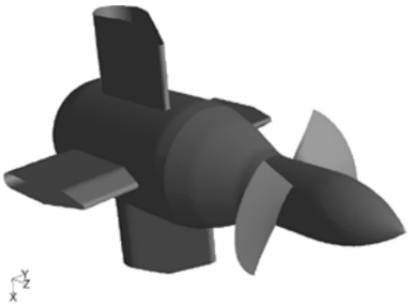

Figure 1. Swirl turbine [2]

Liquid, which flows through draft tube (diffuser), is affected by hydraulic losses. These losses are undesirable and there is an effort to minimize them. The term minimization means process of finding the minimum of examined function. In our case the examined function is coefficient of pressure recovery c_p, defined:

$$c_p = \frac{p_2 - p_1}{\frac{1}{2}\rho v_1^2}, \qquad (1)$$

where p_1 is static pressure at inlet, p_2 is static pressure at outlet; ρ is density of liquid, v_1 is stream velocity at inlet.

2 Nelder-Mead method

Nelder-mead method, known as a simplex method, is a very popular method that does not use derive of the examined functions.

A method is described for the minimization of a function of *n* variables, which depends on the comparison of function values at the *(n+1)* vertices of a general simplex, followed by the replacement of the vertex with the highest value by another point. The simplex adapts itself to the local landscape, and contracts on to the final minimum. The method is shown to be effective and computationally compact [3].

Example of Nelder-Mead method: finding a global minimum of function:

$$f(x_1, x_2) = x_1^2 + x_2^2 - 3x_1 - x_1x_2 + 3\,.$$

Global minimum lies in $x = (2, 1)$. The path of simplex is shown in Figure 2.

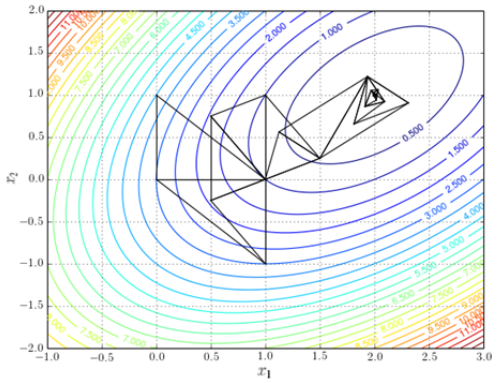

Figure 2. The path of simplex in Nelder-Mead algorithm [4]

[a] Corresponding author: Prokop.Moravec@seznam.cz

2.1 Nelder-Mead algorithm

Nelder-Mead algorithm consists of several main steps [5]:

2.1.1 Choice of initial simplex and parameters

The first step of the algorithm is a creation of regular simplex with side length a, using following definition:

$$x_j = x_1 + pe_j + \sum_{k \neq j} qe_k \,, \tag{2}$$

where x_1 is specified vertex, $\boldsymbol{e_k}$ are unitary basis vectors and

$$p = \frac{a}{n\sqrt{2}}\left(\sqrt{n+1} + n - 1\right);$$

$$q = \frac{a}{n\sqrt{2}}\left(\sqrt{n+1} - 1\right),$$

where n is a dimension. Important part of algorithm is also an ending tolerance criterion:

$$\left(\frac{1}{n+1}\sum_{i=1}^{n+1}[f_i - f(\bar{x})]^2\right)^2 < \epsilon \,, \tag{3}$$

where ε is required precision (tolerance).

Algorithm consists of four basic operations that depend on chosen parameters:

- Reflection $\rho > 0$,
- Expansion $\eta > 1$, $\eta > \rho$,
- Contraction $0 < \gamma < 1$,
- Reduction $0 < \sigma < 1$.

Parameter values are often chosen as follows:

$$\rho = 1;\ \eta = 2;\ \gamma = 0.5;\ \sigma = 0.5 \,.$$

2.1.2 Vertices ordering

The next step involves ordering according to the values of objective function at vertices:

$$f_1 = f(x_1) \leq \cdots \leq f_{n+1} = f(x_{n+1}) \,,$$

after that comes control of tolerance criterion, if epsilon is reached, the end of algorithm will occur.

2.1.3 Reflection

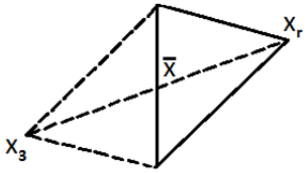

Figure 3. Reflection [5]

First, we must calculate $\bar{x}$ – the centre of gravity of the centroid (Figure 3):

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \,, \tag{4}$$

second, coordinates of reflected vertex x_r are computed (Figure 3):

$$x_r = \bar{x} + \rho(\bar{x} - x_{n+1}) \,, \tag{5}$$

finally, if $f_1 \leq f_r \leq f_n$, then vertex x_r is accepted as replacement of x_{n+1}. Also function value in x_r is calculated $f_r = f(x_r)$.

2.1.4 Expansion

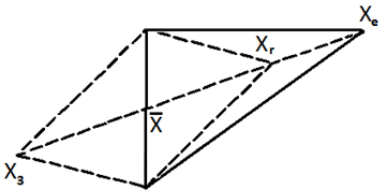

Figure 4. Expansion [5]

Expansion only takes place when $f_r \leq f_1$. Then coordinates of expanded vertex are computed:

$$x_e = \bar{x} + \eta(x_r - \bar{x}) \,, \tag{6}$$

finally, if $f_e < f_r$ then vertex x_e is accepted as replacement of x_{n+1} (else x_r is). Also function value in x_e is calculated $f_e = f(x_e)$.

2.1.5 Contraction

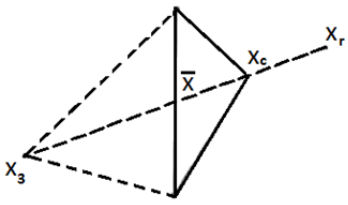

Figure 5. Contraction [5]

Contraction occurs when a given condition $f_r \geq f_n$ is satisfied. Depending on which vertex is better suited (x_r or x_{n+1}), comes external or internal contraction.

Coordinates of internally contracted vertex are computed (if $f_r \geq f_{n+1}$):

$$x_{ci} = \bar{x} + \gamma(x_{n+1} - \bar{x}) \,, \tag{7}$$

then if $f_{ci} < f_{n+1}$, computed vertex x_{ci} is accepted as replacement of x_{n+1}, else coordinates of externally contracted vertex (if $f_r < f_{n+1}$) are computed:

$$x_{ce} = \bar{x} + \gamma(x_r - \bar{x}) \,, \tag{8}$$

then if $f_{ce} < f_r$, computed vertex x_{ce} is accepted as replacement of x_{n+1}. Also function value in x_{ci} or x_{ce} is calculated $f_{ci} = f(x_{ci})$ or $f_{ce} = f(x_{ce})$.

2.1.6 Reduction

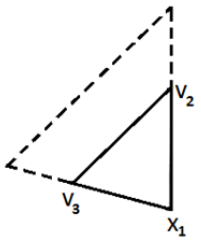

Figure 6. Reduction [5]

In this step of Nelder-Mead algorithm is an edge reduction of initial simplex:

$$v_i = x_1 + \sigma(x_i - x_1) \quad i = 1,..,n+1\,, \tag{9}$$

where v_i, ..., v_{n+1} ($v_1 = x_1$) are vertices of a new reduced simplex (Figure 6.).

Notice: After each step of Nelder-Mead algorithm the control of tolerance criterion must be checked. If is not satisfied, algorithm returns to step called vertices ordering.

3 Adjoint solver

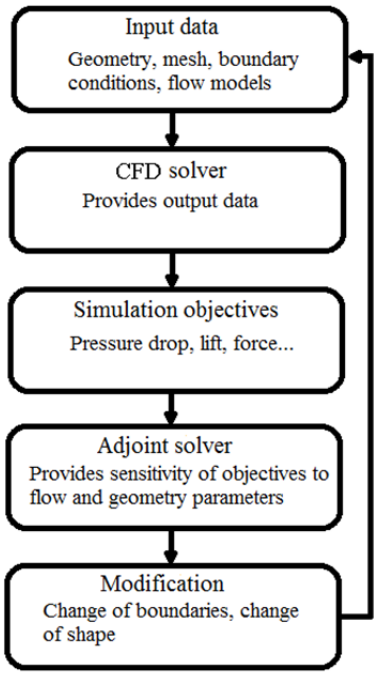

Figure 7. Adjoint solver optimization approach

Adjoint solver is a specialized tool that extends the scope of the analysis provided by a conventional flow solver by providing detailed sensitivity data for the performance of a fluid system [7].

Optimization via adjoint solver consists of several main steps – Figure 7.

There are two types of adjoint approaches – continuous and discrete. Ansys Fluent uses only the discrete one.

3.1 Discrete adjoint approach theory

A detailed description of discrete adjoint approach may be found in [6].

The adjoint solver needs a fully converged flow simulation, from which the gradients of objective function are derived.

The gradient of objective function $J = J[X_D, \mathrm{Q}(X_D)]$ is [6]:

$$G = \frac{dJ}{dX_D} = \frac{\partial J}{\partial X_D} + \frac{\partial J}{\partial \mathrm{Q}}\frac{d\mathrm{Q}}{dX_D}\,, \tag{10}$$

where X_D is vector of design variables (in [6] is reduced to scalar) and fraction $\frac{dQ}{dX_D}$ is referred as flow sensitivities (Q is vector of conserved flow variables).

This gradient could be rewritten in form [6]:

$$G = \frac{\partial J}{\partial X_D} - \psi^T \frac{\partial R}{\partial X_D}\,, \tag{11}$$

where $R = R[X_D, Q(X_D)]$ is the discretized RANS residual vector, ψ represents vector of adjoint variables and is obtained by solving the linear system of equations (known as adjoint equation) [6]:

$$\frac{\partial R^T}{\partial Q}\psi = \frac{\partial J^T}{\partial Q}\,. \tag{12}$$

4 Overview

Initial diffuser design which was used in mathematical optimization methods (Nelder-Mead, adjoint solver) is shown in Figure 8.

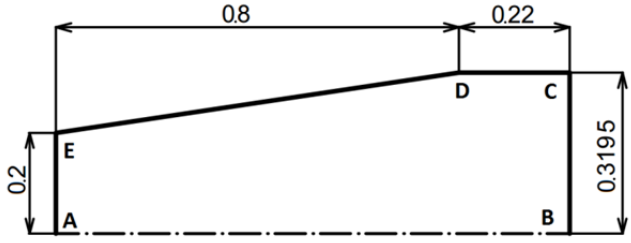

Figure 8. Domain design (dimensions are in metres) [8]

Table 1. displays types of boundary conditions, their locations, magnitudes and turbulence intensity values.

Table 1. Boundary conditions 2D

	Type	Magnitude	Turbulent intensity
A-E	Velocity inlet	2 m/s (constant)	5 %
B-C	Pressure outlet	0 Pa	7 %
A-B	Axis	-	-
E-D-C	Wall	-	-

Notice: Value of velocity at inlet is constant and does not match with real velocity profile behind the runner of swirl turbine.

For initial (Figure 8.) design was created computational mesh (Figure. 9). Value of pressure recovery coefficient for this initial shape diffuser design was **0.736824**.

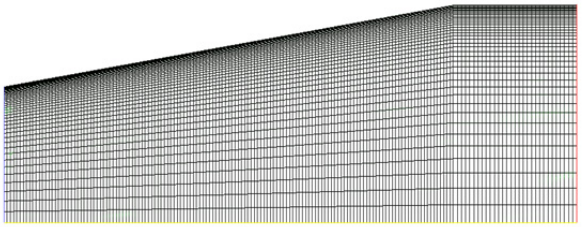

Figure 9. Initial 2D domain mesh

Complete setting of the chosen discretization schemes is listed in the following chart.

Pressure:	Standard
Momentum:	Quick
Turbulent diss. rate:	Second order upwind
Turbulent kinetic energy:	Second order upwind

4.1 Nelder-Mead method [8]

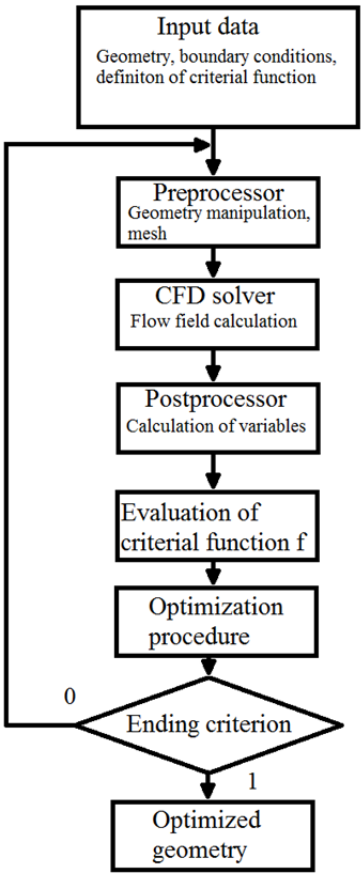

Figure 10. Algorithm of Nelder-Mead optimization [11]

Shape optimization using Nelder-Mead algorithm consists of several steps (Figure 10.).

Very important part is the first step, in which we define the criterial function (c_p in our case), geometry (Figure 8.) and boundary conditions (Table 1.).

Preprocessing involves creation of geometry and computational mesh. Everything is handled in Gambit software using programmed script.

The next two steps – CFD solver and postprocessor – take place in Ansys Fluent, which also uses programmed script. Results (Fluent outcome) are later saved in text files due to easier access to data of examined variables.

The rest of the loop is realized in software called Matlab (including script calling from Gambit and Fluent).

Notice: The geometry change occurs only between vertices *E* and *D*.

4.1.1 Piecewise linear approach

Initial simplex D_i was chosen near vertex *D*. After 75 iterations of Nelder-Mead algorithm, the value of c_p coefficient stabilized at **0.807863**. Coordinates of simplex D_i changed to

$$D_i = [0.0686020; 0.1973931] .$$

Figure 11. shows the change of diffuser geometry (red line).

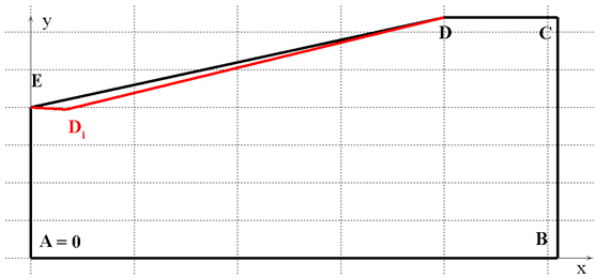

Figure 11. Piecewise linear approach - diffuser geometry

4.1.2 Optimization using Bezier curves

In the next step of optimization, Bezier curves were used as restriction of wall *E-D*.

First order Bezier curve

First order Bezier curve is characterized only by one point of control polygon - P_1.

Number of iterations in this approach was 52. Value of coefficient of pressure recovery c_p reached up to **0.810309**. Coordinates of control point P_1 were computed during Nelder-Mead algorithm as follows:

$$P_1 = [0.138172; 0.185848] .$$

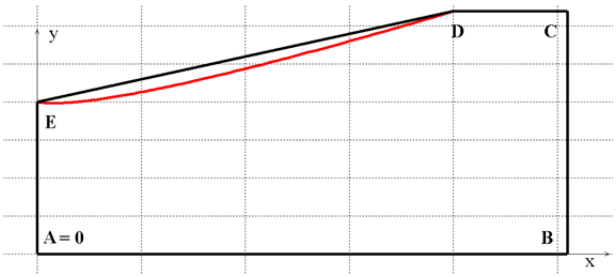

Figure 12. First order Bezier curve – diffuser geometry

Second order Bezier curve

Second order Bezier curve is characterized by two points of control polygon - P_1, P_2.

Number of iterations was 118. Value of coefficient of pressure recovery c_p reached up to **0.812146**. Coordinates of control points P_1 and P_2 were computed during Nelder-Mead algorithm as follows:

$$P_1 = [0.1725671; 0.1814290] ;$$

$$P_2 = [0.5137642; 0.3125861] .$$

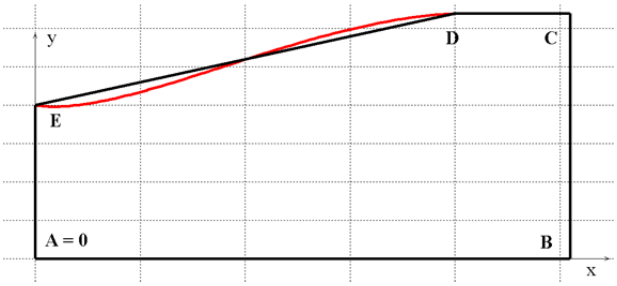

Figure 13. Second order Bezier curve– diffuser geometry

Third order Bezier curve

Third order Bezier curve is characterized by three points of control polygon - P_1, P_2 and P_3.

Number of iterations in this approach was 78. Value of coefficient of pressure recovery c_p reached up to **0.813706**. Coordinates of control points P_1, P_2 and P_3 were computed as follows:

$$P_1 = [0.128563; 0.237429] ;$$

$$P_2 = [0.186375; 0.186554] ;$$

$$P_3 = [0.427714; 0.270078] .$$

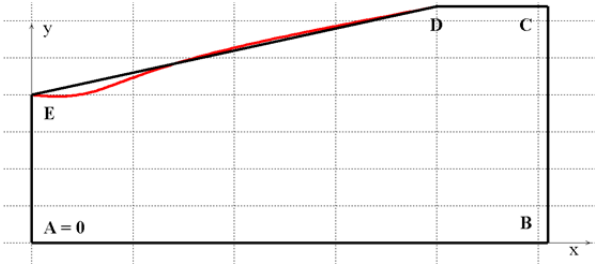

Figure 14. Third order Bezier curve – diffuser geometry

4.2 Adjoint solver

Adjoint solver is incompatible with an axis boundary condition in Fluent. Due to this fact, the working computational domain was created entirely in 3D (Figure 15.). After optimization in 3D, shape of diffuser was transformed back into 2D.

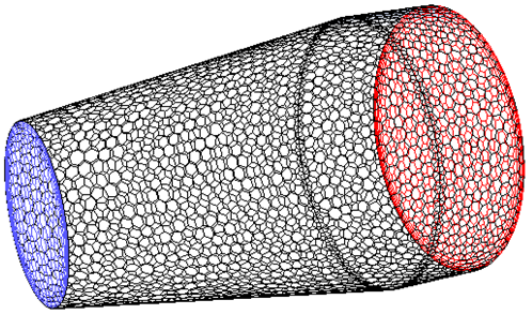

Figure 15. Initial 3D domain mesh

In [10] author recommends polyhedral type of mesh (Figure 16.) to ensure better mesh behaviour during mesh morphing.

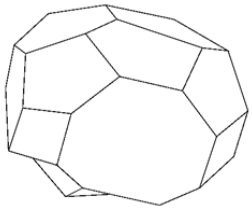

Figure 16. Polyhedral cell [9]

Types of boundary conditions, their locations, magnitudes and turbulence intensity values used in Fluent software during optimization via adjoint solver are the same as in Table 1.

Simulation objective of adjoint solver was set to minimize static pressure at inlet, which is consistent with minimization of c_p.

The next step of adjoint solver optimization approach (Figure 7.) involves shape modification – mesh morphing. Morphing takes place, when sensitivity field is computed (through adjoint equations). This morphing procedure has a double role. Firstly, it smooths the surface sensitivity field and secondly provides smooth distortion in boundary and interior mesh [10]. In adjoint solver rectangular volume encloses boundaries which will be morphed. An array of control points is then distributed in this volume. These points in combination with the local mesh coordinates will define the proper movement both on the boundary and in the interior mesh. The local coordinate system relies on the properties of Bernstein polynomials [10]. Bernstein polynomials are the basis of Bezier curves.

The most important information, which adjoint solver offers are optimal displacements (Figure 17.). Vectors of optimal displacement show location and intensity of diffuser shape change.

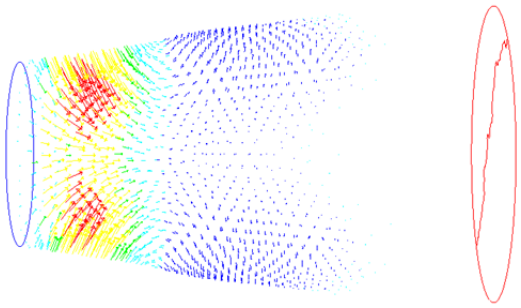

Figure 17. Optimal displacements - vectors

After 30 loops optimization was stopped, 3D geometry was transformed back to 2D and 2D geometry was recalculated using the same CFD settings as in Nelder-Mead algorithm. Coefficient of pressure recovery in this case reached up to value **0.812354** (22. loop – Figure 18.).

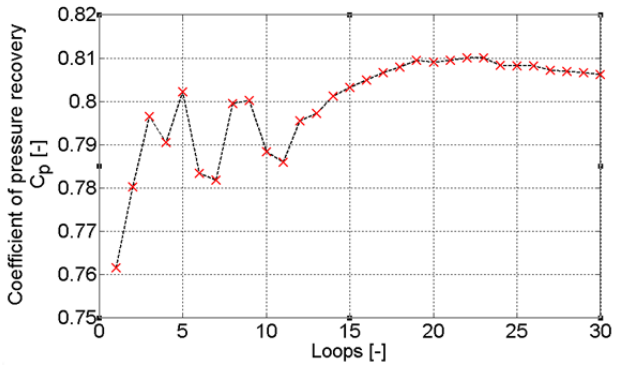

Figure 18. Adjoint solver approach

Due to the absence of better restriction of shape (for example: using only the first order Bezier curve as a wall restriction) vertices of changed wall were later approximated in Excel using the 4. order polynomial (Figure 19.). Coefficient of pressure recovery changed to **0.812614**.

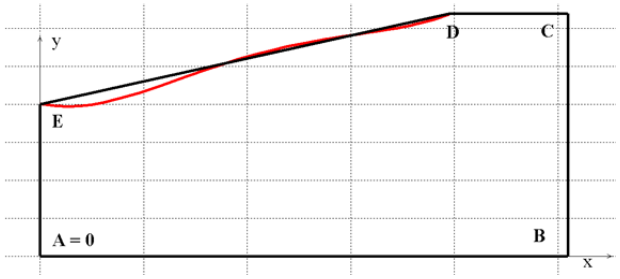

Figure 19. Adjoint solver – 4. order polynomial approx. – diffuser geometry

5 Conclusions

The values of c_p obtained in both optimization methods – Nelder-Mead and adjoint solver - are summarized in Table 2.

Optimized shape of diffuser via Nelder-Mead method has 10.43 % higher pressure recovery than the original (initial) shape. This value was achieved through the use of 3. order Bezier curve as a wall constraint. The benefits of this method include: achieved value of c_p and ability to control changed boundaries. Disadvantages include: the need to use additional software (Matlab) and the higher number of loops to achieve the final value of c_p.

Optimized shape of diffuser via adjoint solver in Fluent has 10.25 % higher pressure recovery than the original (initial) shape, but this shape had extensively corrugated boundary. Problem was solved by polynomial approximation of vertices that form the diffuser. Value of c_p increased by 0.04 %. The main advantages of this method are: the implementation in Ansys Fluent and shape modification is also present in the solver. Disadvantages include mainly the absence of better restriction of changed boundaries and a large number of settings, which adjoint solver uses.

Comparison of execution time for both methods has not been done due to use of two different computers (clusters).

It should be also mentioned that the value of velocity at inlet was constant and does not correspond with the real velocity profile behind the runner of swirl turbine.

Coefficient of pressure recovery c_p diffuser increases as a result of weaker velocity gradients and shear layers (it was also confirmed by theoretical analysis [11]).

Table 2. Overview

	c_p [-]	Change [%]
Initial geometry	0.736824	-
Nelder-Mead – Piecewise linear approach	0.807863	9.64
Nelder-Mead – Bezier curve 1. order	0.810309	9.97
Nelder-Mead – Bezier curve 2. order	0.812146	10.22
Nelder-Mead – Bezier curve 3. order	0.813706	10.43
Adjoint solver	0.812354	10.25
Adjoint solver – 4. order polynomial approximation	0.812614	10.29

References

1. F. Pochylý, M. Haluza, P. Rudolf, F. Šob: Design of a New Low-Head Turbine. In Proceedings of the XXIst Symposium on Hydraulic Machinery and Systems. September 9-12. 2002. Lausanne. pp. 29-34. ISBN 3-85545-865-0
2. P. Rudolf: Optimization Methods for Hydraulic Machines Design - Shape Optimization of Swirl Turbine Draft Tube. Conference paper – Hydroturbo. September 2006.

3. J.A. Nelder, R. Mead: A Simplex Method for Function Minimization. 1965. The Computer Journal. 7(4): 308-313. DOI: 10.1093/comjnl/7.4.308. ISSN 0010-4620. Available online: http://comjnl.oxfordjournals.org/cgi/doi/10.1093/comjnl/7.4.308
4. Nelder-mead simplex algorithm. Figure: one-minimum.png [online]. [cit. 2015-09-01]. Available online: http://www.jakubkonka.com/images/nelder-mead/one-minimum.png
5. J. Machalová, H. Netuka: Numerické metody nepodmíněné optimalizace. 1. vyd. Olomouc: Univerzita Palackého v Olomouci, 2013, 142 s. ISBN 978-80-244-3403-2
6. M. Nemec, D.W. Zingg: Newton - Krylov Algorithm for Aerodynamic Design Using the Navier-Stokes Equations, AIAA Journal, Vol. 40, No. 6, June 2002, pp. 1146-1154.
7. ANSYS FLUENT 16.1 Manual
8. J. Hliník: Shape optimization of hydraulic turbine diffuser. Bachelor thesis. 2015, Brno.
9. Converting the domain to a polyhedra. Figure: img474.gif [online]. [cit. 2015-09-01]. Available online: https://www.sharcnet.ca/Software/Fluent6/html/ug/img474.gif
10. A. Tzanakis: Duct optimization using CFD software 'ANSYS Fluent Adjoint Solver'. Master's thesis in Automotive Engineering. Department of Applied Mechanics. Division of Vehicle Engineering and Autonomous Systems. Chalmers University of Technology. Goteborg, Sweden 2014
11. P. Rudolf: Study of the shear layers for swirl draft tube optimization. PhD thesis. In Czech

Acknowledgement

Presented research was supported by centre of competence of Technology Agency of the Czech Republic TE02000232 “Rotary machines”.