Computer Assisted Mechanics and Engineering Sciences, **17**: 147–160, 2010.

Numerical investigations of the convergence of a remeshing algorithm on an example of subsonic flow

J. Kucwaj
Cracow University of Technology
Institute of Computer Science
Department of Physics, Mathematics and Computer Science
Warszawska 24, 31-155 Cracow, Poland
e-mail: jkucwaj@pk.edu.pl

The main goal of the paper is to analyze convergence of a remeshing scheme evaluated by the author [8] on the example of a potential flow around a profile. It is assumed that flow is stationary, irrotational, inviscid and compressible. The problem is led to solving nonlinear differential equation with additional nonlinear algebraic equation representing the so called Kutta-Joukovsky condition. For adaptation a remeshing scheme is applied. For every adaptation step mesh is generated using grid generator [7], which generates meshes with mesh size function. The mesh size function is modified at every adaptation step by nodal values of the error indicator interpolation. The nonlinear algebraic system of equations obtained from discretizing of the problem, is solved by the application of the Newton-Raphson method.

Keywords: finite element method, fluid mechanics, grid generation, remeshing, Kutta-Joukovsky condition.

1. Introduction

The main goal of this paper is to solve a subsonic flow problem by an adaptive method, based on numerical grid generator with mesh size function [4]. The background of the method was presented by the author in [8]. It is assumed that flow is stationary, irrotational, inviscid and compressible. For the unique problem solution the so called Kutta-Joukovsky condition [5] is taken. The problem is led to solve a nonlinear differential equation with a nonlinear algebraic equation, given by the additional Kutta-Joukovsky condition. The solution method belongs to the wide class of adaptation techniques of the remeshing type [9].

The problem is carried to weak formulation [1], which is equivalent to the search for stationary point of a functional. After discretisation, the problem is carried to solve a system of nonlinear algebraic equations. The system is solved by the Newton-Raphson method. The Kutta-Joukovsky condition is used to find the unknown jump β of potential function along slit Σ (Fig. 1).

The method is based on error analysis and then, taking into account the error indicator, a new mesh for finite element method is generated in the whole domain, in order to decrease value of the error. Those techniques are other than those presented in [3, 10], where only elements with greatest errors are divided into smaller without mesh modification in the whole domain.

The proposed algorithm assumes that there is given an initial mesh size function used for mesh generation, which is used to solve the problem. Then the error indicator is found at the nodes of the mesh. The nodal values of the error are transformed throughout linear mapping, described by the coefficients indicating the gradation of the mesh size function in the way, that in zones of greatest error there is the greatest mesh size value reduction, and the dependence is invertibly proportional. The obtained nodal values are used as interpolation values to define modified mesh size function.

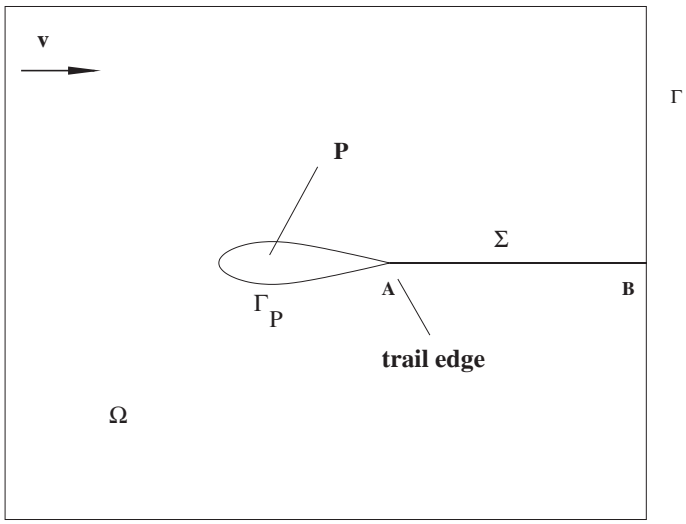

Fig. 1. Computational domain.

The obtained mesh size function is used for a new mesh generation. By repeating the procedure, the mesh is adapted to get better solution.

The secant method is applied to satisfy Kutta-Joukovsky condition. 8–10 iterations of secant method gave error of the order 10^{-9}.

2. Problem formulation

It is assumed that the flow is stationary, irrotational, compressible, inviscid in domain Ω around the profile P (Fig. 1). The following notations are used:

Γ_P – the boundary of profile P,

Σ – the slit from **A** to point **B**,

Γ_∞ – external component of the boundary Ω.

The boundary of Ω consists of the following parts:

$$\partial\Omega = \Gamma_\infty \cup \Gamma_P, \tag{1}$$

which interiors do not intersect. The curve Γ_∞ is an artificially taken boundary to obtain bounded computational domain. The boundary Γ_P is the border of the contact with the obstacle, which has vertex modeling trail edge [1]. Unbounded domain besides Γ_∞ is introduced as

$$\Omega^c = \mathbb{R}^2 \setminus (\overline{\Omega \cup P}). \tag{2}$$

Summarizing, the formulation of the boundary value problem is as follows:

$$div[\rho(|\nabla u|^2)\nabla u] = 0, \quad \text{in} \quad \Omega, \tag{3}$$

where u is the speed potential $v = \nabla u$ and

$$\rho(|\nabla u|^2) = \rho_0 \left(1 - \frac{\kappa - 1}{2a_0^2}|\nabla u|^2\right)^{\frac{1}{\kappa - 1}} \tag{4}$$

is the gas density.

Here $\kappa > 1$ is the adiabatic gas constant, e.g. $\kappa = 1.4$ for dry air. The constant ρ_0, a_0 are the density and the local speed of sound, respectively, for the motionless gas.

3. Discretisation method

The full potential flow problem is formulated as follows:

$$div[\rho(|\nabla u|^2)] = 0 \quad \text{in} \quad \Omega, \qquad \frac{\partial u^+}{\partial \mathbf{n}} - \frac{\partial u^-}{\partial \mathbf{n}} = 0 \quad \text{on} \quad \Sigma, \tag{5}$$

where u^+ and u^- are values of u, respectively, over upper and lower part on the slit, and

$$u^+ - u^- = \beta \quad \text{for some jump } \beta \text{ on} \quad \Sigma, \tag{6}$$

$$\rho(|\nabla u|^2)\frac{\partial u}{\partial \mathbf{n}} = \rho(|\mathbf{u}_\infty|^2)\mathbf{v}_\infty \mathbf{n}_\infty \quad \text{on} \quad \Gamma_\infty, \tag{7}$$

where $\mathbf{n}_\infty$ is the external normal to Γ_∞, and $\rho_\infty = \rho(|\mathbf{v}_\infty|)$.

To determine the jump β, we need some additional Kutta-Joukovsky condition

$$K(\beta) = |\nabla u^+|^2_{\mathbf{A}} - |\nabla u^-|^2_{\mathbf{A}} = 0, \tag{8}$$

at the trail edge.

For the sake of the weak formulation the following functional spaces are introduced:

$$W^{1,p}(\Omega) = \left\{ v \in L^p(\Omega), \quad \frac{\partial v}{\partial x_i} \in L^p(\Omega), \quad i = 1,2 \right\}, \tag{9}$$

$$W^{1,p}(\dot{\Omega}) = \left\{ v \in L^p(\dot{\Omega}), \quad \frac{\partial v}{\partial x_i} \in L^p(\dot{\Omega}), \quad i = 1,2 \right\}, \tag{10}$$

$$W^{1,\infty}(\dot{\Omega}) = \left\{ v \in L^\infty(\dot{\Omega}), \quad \frac{\partial v}{\partial x_i} \in L^\infty(\dot{\Omega}), \quad i = 1,2 \right\}. \tag{11}$$

Multiplying (5) by arbitrary trial function v, integrating by parts and taking into consideration the Neumann boundary conditions the following weak formulation is obtained:

$$\iint\limits_{\Omega} \rho(|\nabla u|)\nabla u \nabla v dx\, dy = \int\limits_{\Gamma} h v d\Gamma, \quad v \in W^{1,\infty}(\dot{\Omega}), \tag{12}$$

$$u^+ - u^- = \beta \quad |\nabla u^+|^2_{\mathbf{A}} - |\nabla u^-|^2_{\mathbf{A}} = 0, \tag{13}$$

where

$$h = \rho_\infty \mathbf{v}_\infty \mathbf{n}_\infty \quad \text{on} \quad \Gamma_\infty. \tag{14}$$

The weak formulation (12)–(14) can be equivalently formulated as a search for the extremum of the following functional:

$$I(u) = \frac{1}{2}\iint\limits_{\dot{\Omega}} R(|\nabla u|) dx dy - \int\limits_{\Gamma} h v d\Gamma, \tag{15}$$

where

$$R(s) = \int\limits_0^s \rho(s) ds. \tag{16}$$

Equation (5) is directly obtained from Euler's equations of variational calculus applied to (15). By substituting

$$F(x,y,u,u_x,u_y)=\frac{1}{2}R(|\nabla u|^2),$$

it can be written:

$$I(u)=\int\limits_{\dot{\Omega}} F(x,y,u,u_x,u_y)d\dot{\Omega}-\int\limits_{\Gamma} hvd\Gamma, \tag{17}$$

where $\Omega\subset\mathbb{R}^2$ is a domain of the variational problem. For the finite element solution to the problem, grid $\mathcal{T}_0$ is generated with the given, positively defined mesh size function: $\gamma_0:\Omega\mapsto\mathbb{R}$. Then the approximation space is defined as:

$$V^0=\{v:\bigcup_{i=1}^{n_0}\overline{T}_i^0\mapsto\mathbb{R}, v \ \text{ continuous, } v|_T\in P_1,\ \forall T\in\mathcal{T}_0\}, \tag{18}$$

where $\mathcal{T}_0=\{T_i:i=1,\dots n_0\}$ is the set of non-intersecting triangles covering the domain.

4. Calculation of the tangent matrix

Let's introduce the finite element basis $\{U_i\}_{i=1}^N$ in the space V^0 the discretisation of the problem leads to the solution of the following system of nonlinear algebraic equations:

$$g_k(\lambda_1,\lambda_2,\dots,\lambda_N)=\frac{\partial I(\sum_{i=1}^N\lambda_iU_i)}{\partial\lambda_k}=0,\qquad \text{for}\ \ k=1,\dots,N, \tag{19}$$

where $\lambda_1,\lambda_2,\dots,\lambda_N$ are coefficients of the linear combination of the approximate solution.

The chain formula for partial derivatives leads to:

$$\begin{aligned} g_k(\lambda_1,\lambda_2,\dots,\lambda_N)&=\frac{\partial}{\partial\lambda_k}\int\limits_{\Omega}F(x,y,\sum_{i=1}^N\lambda_iU_i,\sum_{i=1}^N\lambda_iU_{i,x},\sum_{i=1}^N\lambda_iU_{i,y})d\Omega-\int\limits_{\Gamma}gU_kd\Gamma\\ &=\int\limits_{\Omega}D_F^T\Psi_kd\Omega-\int\limits_{\Gamma}gU_kd\Gamma, \end{aligned} \tag{20}$$

and

$$\frac{\partial g_k}{\partial\lambda_j}=\int\limits_{\Omega}\Psi_k^TD_{FF}\Psi_jd\Omega. \tag{21}$$

In formulas (20), (21)

$$\mathbf{D}_F=\left[\frac{\partial F}{\partial u},\quad\frac{\partial F}{\partial u_x},\quad\frac{\partial F}{\partial u_y}\right]^T, \tag{22}$$

$$\mathbf{D}_{FF}=\begin{bmatrix} \dfrac{\partial^2F}{\partial u\partial u}, & \dfrac{\partial^2F}{\partial u\partial u_x}, & \dfrac{\partial^2F}{\partial u\partial u_y}\\ \dfrac{\partial^2F}{\partial u_x\partial u}, & \dfrac{\partial^2F}{\partial u_x\partial u_x}, & \dfrac{\partial^2F}{\partial u_x\partial u_y}\\ \dfrac{\partial^2F}{\partial u_y\partial u}, & \dfrac{\partial^2F}{\partial u_y\partial u_x}, & \dfrac{\partial^2F}{\partial u_y\partial u_y} \end{bmatrix}, \tag{23}$$

and

$$\mathbf{U} = \begin{bmatrix} U_1, & \ldots, & U_N \\ U_{1,x}, & \ldots, & U_{N,x} \\ U_{1,y}, & \ldots, & U_{N,y} \end{bmatrix}. \tag{24}$$

Presented formulas are valid for arbitrary basis functions (for example polynomial) $\{U_1, U_2, \ldots, U_N\}$ defined in Ω.

In case of FEM we have the following formulas:

$$g_j = \int_{\Omega} \mathbf{D}_F^T \psi_j \, dx \, dy = \sum_{e=1}^{N_T} \int_{T_e} \mathbf{D}_F^T \psi_j \, dx \, dy, \tag{25}$$

$$\frac{D(g_1, g_2, \ldots, g_N)}{D(\lambda_1, \lambda_2, \ldots, \lambda_N)} = \frac{\partial g_i}{\partial \lambda_j} = \sum_{e=1}^{N_T} \int_{T_e} \psi_i^T \mathbf{D}_{FF} \psi_j \, dx \, dy, \tag{26}$$

where ψ_k is the *k-th* column of matrix $\mathbf{U}$.

Introduce the following vector:

$$\mathbf{G}(\mathbf{\Lambda}) = [g_1(\mathbf{\Lambda}), \ldots, g_N(\mathbf{\Lambda})]^T, \tag{27}$$

and matrix

$$\mathbf{J_G} = \begin{bmatrix} \dfrac{\partial g_i}{\partial \lambda_j} \end{bmatrix}. \tag{28}$$

Let the set $\{u_1^e, u_2^e, \ldots, u_{n_e}^e\}$ will be a set of shape functions of the element with number e for $e = 1, \ldots, N_T$, then the matrix $\mathbf{u}_e$ is defined as follows:

$$\mathbf{u}_e = \begin{bmatrix} u_1, & \ldots, & u_{n_e} \\ u_{1,x}, & \ldots, & u_{n_e,x} \\ u_{1,y}, & \ldots, & u_{n_e,y} \end{bmatrix}, \tag{29}$$

where n_e is the number of the shape function of *e-th* element. Introduce the vector $\mathbf{g}_e$ and the matrix $\mathbf{A}_e$

$$\mathbf{g}_e := \mathbf{D}_F^T \mathbf{u}_e, \qquad \mathbf{A}_e := \mathbf{u}_e^T \mathbf{D}_{FF} \mathbf{u}_e. \tag{30}$$

The considered problem in the form of a search for a extremum of the functional (17), allows to perform a general computer code, giving the solution possibility of class of problems, which can be carried to such a formulation. The only replacement routines would be routines calculating vector $\mathbf{D}_F$, matrix $\mathbf{D}_{FF}$ and starting vector Λ for Newton-Raphson iteration.

5. Application of Newton-Raphson method to the solution to the nonlinear algebraic system of equations

To solve the system of nonlinear equations the Newton-Raphson method is applied. The vector $\mathbf{G}$ and matrix $\mathbf{J_G}$ depend on $\mathbf{\Lambda}$. The Newton-Raphson method consists of the following steps:

1. Fix initial vector $\mathbf{\Lambda}_0$, set i=0;
2. repeat points 3, 4, 5 until $||\mathbf{G}(\mathbf{\Lambda_i})|| < \epsilon \, ||\mathbf{\Lambda_i}||$;

3. solve the following system of linear equations:

$$\mathbf{J_G}(\mathbf{\Lambda}_i)\mathbf{\Delta\Lambda}_{i+1} = -\mathbf{G}(\mathbf{\Lambda}_i); \tag{31}$$

4. $\mathbf{\Lambda}_{i+1} = \mathbf{\Lambda}_i + \mathbf{\Delta\Lambda}_{i+1}$;

5. $i := i + 1$.

It is assumed, that the norm in $\mathbb{R}^N$ is defined as:

$$||\mathbf{x}|| = \max_{i=1,\ldots,N} |x_i|, \qquad \text{where} \qquad \mathbf{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N. \tag{32}$$

The sequence of vectors $\mathbf{\Lambda}_0, \mathbf{\Lambda}_1, \ldots$ tends to the solution. At every iteration step the Jacobi matrix must be assembled. In the presented examples usually 10–15 iterations were sufficient to obtain the value $||\mathbf{G}(\mathbf{\Lambda}_i)||$ of residuum norm of order 10^{-9}.

6. The unstructured grid generation with mesh size function over arbitrary domains with curved boundaries

Grid generation with arbitrary size is performed by the 2-D generator [4]. The main idea of grid generation is based upon the algorithm of the advancing front technique, and generalization of Delaunay triangulation for arbitrary domains. It is assumed, that the domain is multiconnected with arbitrary numbers of internal loops. The boundary of the domain may be composed of the following curves:

- straight line segment,
- arc of circle,
- B-spline curve.

In case of the advancing front technique combined with Delaunay triangulation, the point insertion and triangulation can be divided into the following steps:

1. points generations on the boundary components of the boundary of the domain,
2. internal points generation by the advancing front technique,
3. Delaunay triangulation of the previously obtained set of points,
4. Laplacian smoothing of the obtained mesh.

The algorithm for boundary points generation depends upon the type of the boundary segment.

6.1. Points generation on *straight-line* segment

For interpolation of the mesh size function the segment is divided into equidistance length intervals. For simplicity of used formulas, for points insertion on the considered part of the curve, the piecewise linear interpolation is applied. The algorithm of points generation in the considered segment is recursive [8]:

- first point is one of the ends of the considered curve,

- provided that, the point **A** is just generated and **B** the next point to be inserted. The point **B** should satisfy the following mesh density function requirement:

$$dist(\mathbf{A},\mathbf{B}) = \frac{\gamma(\mathbf{A}) + \gamma(\mathbf{B})}{2}, \tag{33}$$

where $\gamma = \gamma(\mathbf{X})$ is the mesh size function. Taking into account piece-wise linear interpolation of γ and equation (33) the point **B** is found. Additionally, it is necessary to find out, to which one of the interpolation intervals, the searched point **B** should belong to,

- the process continues till the last generated point **A** satisfies the condition:

$$\gamma(\mathbf{A},\mathbf{B}) > \frac{1}{2} dist(\mathbf{A},\mathbf{C}), \tag{34}$$

where **C** is the second endpoint of the curve segment.

6.2. Generation of points on arc of circle

A circle, as every piece of the boundary curve, is defined by its ends and, additionally, by the center of the circle containing the arc. The following notations are taken:

S – centre of the circle,

R – radius of the circle,

A, B – ends of the arc.

The algorithm of points generation on the arc consists of the following points:

1. The arc is mapped onto the interval, the length of which equals to the length of the arc,
2. A new mesh size function is defined by using the size function given on the arc,
3. Points generation on the interval with the newly given size function,
4. Remapping the obtained nodes onto the curve.

The interval on which the arc is mapped is equal to $[0, M]$, where:

$$M = |\ \overset{\smile}{\mathbf{AB}}\ |, \tag{35}$$

where $\overset{\smile}{\mathbf{AB}}$ is an arc with ends $\mathbf{A}, \mathbf{B}$. The mapping of the arc onto the interval is defined as follows:

$$K : \overset{\smile}{\mathbf{AB}} \longmapsto [0, M], \tag{36}$$

where for every point $\mathbf{P} \in \overset{\smile}{\mathbf{AB}}$, $K(\mathbf{P}) = |\ \overset{\smile}{\mathbf{AP}}\ |$. The mesh size function on the interval $[0, M]$ is defined as follows:

$$\gamma_M(x) := \gamma(K^{-1}(x)). \tag{37}$$

The points on the interval $[0, M]$ are generated by the technique presented in the previous section. The obtained points in the interval $[0, M]$ are mapped into the arc by using the inverse mapping K^{-1}.

6.3. Points generation on the B-spline curve

A B-spline curve is defined by a set of control vertices $\mathbf{V_0}, \mathbf{V_1}, \ldots, \mathbf{V_m}$, (see Fig. 2) and the formula below. The set of control vertices forms the control polygon. The B-spline curve is defined by the following formula:

$$\mathbf{C}(t) = \sum_{i=0}^{m} \mathbf{V}_i B_{i,p}(t), \tag{38}$$

where $B_{i,p}$ are B-spline functions of order p, defined by the Cox-de'Boor [4] recursive formula:

$$B_{i,p}(t) = \frac{(t - t_i)B_{i,p-1}(t)}{t_{i+p-1} - t_i} + \frac{(t_{i+k} - t)B_{i+1,p-1}(t)}{t_{i+p} - t_{i+1}}, \tag{39}$$

for $p > 0$, and

$$B_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1}, \\ 0 & \text{otherwise,} \end{cases}$$

if $p = 0$.

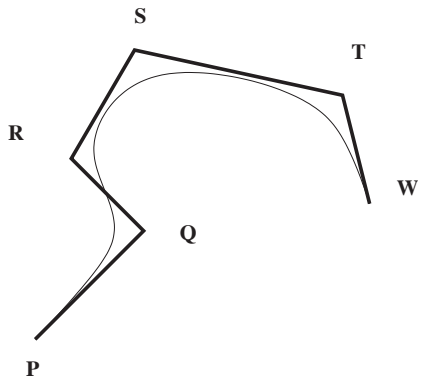

Fig. 2. Control polygon.

B-spline functions are defined on the following knot vector [4]:

$$\mathbf{V} = \{\underbrace{t_0, \ldots, t_0}_{\text{p+1 times}}, t_1, \ldots, t_n, \underbrace{t_{n+1}, \ldots, t_{n+1}}_{\text{p+1 times}}\}, \tag{40}$$

where

$$t_0 < t_1 < \ldots t_n < t_{n+1}, \quad \text{and} \quad m = n + 2p + 1. \tag{41}$$

The point generation on the B-spline curve begins with points generation on the control polygon, and then the generated nodes are projected onto the B-spline curve to obtain the final grid. The projection is done by using parametric equation of the curve, where points at first are generated on the interval, and then mapped onto the curve. As the knot vector for B-spline definition, the control polygon mapped onto straight line is taken in such a way, that the straight-line segments remain "stiff", the points $\mathbf{V_0}, \mathbf{V_1}, \ldots, \mathbf{V_m}$ are treated as wrists, and the control polygon is "straighten".

The proposed idea may be summarized as follows:

1. Define an interval $[0, M]$, where:

$$M = \sum_{i=0}^{m-1} |\mathbf{V_i V_{i+1}}|. \tag{42}$$

Let $\mathbf{C}^*$ denotes the control polygon defined by $\mathbf{V}_i$, $i = 0, 1, \ldots, m$.
Then the mapping:

$$K : \mathbf{C}^* \mapsto [0, M], \tag{43}$$

is defined as follows: let $\mathbf{P} \in C^*$ then

$$K(\mathbf{P}) = \sum_{i=0}^{k-1} |\mathbf{V}_i\mathbf{V}_{i+1}| + |\mathbf{P}\mathbf{V}_{k+1}|, \tag{44}$$

where k is such that $\mathbf{P} \in \mathbf{V}_k\mathbf{V}_{k+1}$. It is easy to show, that $K(\mathbf{P}) \in [0, M]$.

2. The knot vector [4]:

$$\mathbf{V} = \{\underbrace{0, \ldots, 0}_{\text{p+1, times}}, t_1, \ldots, t_n, \underbrace{t_{n+1}, \ldots, t_{n+1}}_{\text{p+1 times}}\}, \tag{45}$$

where

$$0 < t_1 < \ldots t_n < t_{n+1},$$

is defined as:

$$t_i = K(\mathbf{V}_i), \ for \ i = 0, 1, \ldots, n+1.$$

3. Assume that the mesh size function:

$$\gamma : D \mapsto \mathbb{R}, \tag{46}$$

is given, where D is a 2-D domain containing B-spline curve $\mathbf{C}$, and control polygon C^*. Then the mesh size function:

$$\gamma_M : [0, M] \mapsto \mathbb{R} \tag{47}$$

is defined by:

$$\gamma_M(t) = \gamma(\mathbf{C}(t)) \quad \text{where} \quad t \in [0, M]. \tag{48}$$

Finally, the points on the interval $[0, M]$ are generated using the algorithm previously defined. In that way a set of points $\mathbf{P}_0, \ \mathbf{P}_1, \ldots, \mathbf{P}_l$ is obtained.

4. Points $\mathbf{P}_i$ $i = 0, 1, \ldots, l$ are mapped onto the B-spline curve $\mathbf{C} = \mathbf{C}(t)$ by using the equation (38). It is recommended to project orthogonally the obtained points on the control polygon, and then again onto interval [0,M] by using transformation $K(\mathbf{P})$. In such a way a new set of points $\mathbf{P}_i^{'}$, $i = 0, 1, \ldots, l$ is transformed to the B-spline curve.

In the presented algorithm it is assumed, that the control polygon is close enough to the B-spline curve. In case of polygonal line $K(\mathbf{P})$ is equal to the natural parameter of the curve, thus in case of arbitrary curve the proposed approach is only the approximation of the natural parameter.

6.4. Generation of internal points

As in the proposed approach the advancing front technique (the Delaunay advancing front technique [9]) is applied to the triangulation of the domain, that same approach is applied to the internal points generation. In this paper the idea of internal points generation and afterwards Delaunay triangulation is preferred. Over a considered straight-line segment a new point is found as the intersection of two circumcircles with centers at the segment ends and with the radiuses equal, appropriately, to values of the mesh size function at those ends. Except of that the standard advancing front [7] approach is implemented.

7. Algorithm of remeshing

The whole algorithm of adaptation is realized in successive generation of sequence of meshes $\{\mathbf{T}_\nu\}$, where $\nu = 0, 1, 2, \ldots$ with modified mesh size function. By using every mesh of the sequence, the problem is solved, and then appropriate error indicator at every element is obtained. The value of the error indicator is reduced to the nodes by the averaging method. Having values of errors at nodes, a continuous error function in the whole domain is constructed, by using piecewise linear interpolation at all the elements. The error function is appropriately transformed to obtain a multiplier for mesh size function.

The proposed approach gives us the possibility of to solve to the considered problem on well-conditioned meshes and to obtain optimal graded meshes.

7.1. Remeshing Scheme

The algorithm of remeshing can be divided into the following steps:

1. preparation of the information about the geometry and boundary conditions of the problem to be solved,
2. fixing an initial mesh size function,
3. mesh generation with the mesh size function,
4. solution to the problem 19 on the generated mesh,
5. evaluation of error indicator in every element,
6. calculation of nodal error indicator values by using averaging method,
7. definition of the new mesh size function by using the errors found at every point,
8. if error not satisfactory go to point 3.,
9. end of computations.

In the examples solved by the author of this paper it was sufficient to make from 3 to 7 steps of adaptation.

7.2. Error indicators

The applied indicators are calculated for every element or directly at the nodes [4, 9]:
Let e_i for $i = 1, \ldots, n_0$ be an error indicator at *i-th* apex of the mesh $\mathcal{T}_0$, and $\mathcal{P}_0 = \{P_i,\ i = 1, \ldots, n_P\}$ – set of nodes. We define a patch of elements for every node P_i as:

$$L_i = \{k : P_i \in \overline{T}_k\} \text{ for } i = 1, \ldots, n_P. \tag{49}$$

1. Let N_i be a set of neighbors of *i-th* element e.q.:

$$N_i = \{k :\ T_k \quad \text{has a common edge with} \quad T_i\}, \tag{50}$$

then

$$\widetilde{e}_i = \sqrt{\sum_{k \in N_i} \left(\frac{\partial u_i}{\partial n_k} - \frac{\partial u_k}{\partial n_k}^2 \right)^2}, \tag{51}$$

where u_i is the restriction of the solution to *t-th* element and n_k is unit normal the edge common with *k-th* and *i-th* element.

2. In this case it is suggested to introduce directly values of error indicator at every node of the mesh. The error indicator is suggested by the author. From the numerical analyses it follows, that the application of this error indicator causes generation similar meshes to the firstly defined.

$$e_i = \sqrt{\sum_{k\in L_i,\ l\in L_i,\ l\neq k} \left(\frac{\partial u_i}{\partial x} - \frac{\partial u_k}{\partial x}^2\right)^2 + \left(\frac{\partial u_i}{\partial y} - \frac{\partial u_k}{\partial y}^2\right)^2}, \tag{52}$$

where L_i is the set of numbers of elements meeting at i-th node.

7.3. Modification of the mesh size function

The modification of the mesh size function is performed at every adaptation step for the realization of the next one. The main idea of this part of the algorithm relies on reduction of the values of the mesh size function by an appropriately chosen function. The chosen function is continuous, linear and has the smallest value at node where the value of the error indicator is maximal and the greatest where the value of the error is minimal. It increases when error decreases. To describe the algorithm of the mesh size function modification it is necessary to reduce the values of the error to a node. For every node P_i a weighted averaged value of the indicator is defined as follows:

$$\widetilde{e}_i = \frac{\sum_{k\in L_i} area(T_k)e_k}{\sum_{k\in L_i} area(T_k)}, \tag{53}$$

where $L_i = \{k:\ P_i \in T_k\}$ and T_k is the k-th element.

In such a way a set of values of the error at every nodal point is given.

$$\alpha = \min_{k=1,2,\ldots,N_{NOD}} \widetilde{e}_k, \qquad \beta = \max_{k=1,2,\ldots,N_{NOD}} \widetilde{e}_k, \tag{54}$$

where N_{NOD} is the number of nodes. Obviously, $\alpha \leq \widetilde{e}_k \leq \beta$ for $k = 1, \ldots, N_{NOD}$.

The following new values are introduced:

λ – a value indicating the greatest mesh size function reduction,

μ – a value indicating the smallest mesh size function reduction.

Usually λ and μ have positive values less than 1, and additionally $\mu < \lambda$. Define the transformation

$$l:\ [\alpha, \beta] \mapsto [\mu, \lambda] \tag{55}$$

satisfying conditions: $l(\alpha) = \lambda$ and $l(\beta) = \mu$. By these assumptions $\mu \leq l(x) \leq \lambda$. Provided that

$$Q_i = l(\widetilde{e}_i) \quad \text{for} \quad i = 1, \ldots, N_{NOD}, \tag{56}$$

then we have: $\min_{i=1,2,\ldots,N_{NOD}} Q_i = \mu$, $\max_{i=1,2,\ldots,N_{NOD}} Q_i = \lambda$.

Introducing the function $r:\ \overline{D} \mapsto I\!R$ as follows: $r(\overline{x}) = \Pi(\overline{x})$ if $\overline{x} \in \overline{T}_s$, where Π is an affine mapping of two variables satisfying the following conditions:

$$\Pi(P_i) = Q_i \quad \text{for} \quad i = 1, 2, 3, \tag{57}$$

where P_1, P_2, P_3 are the vertices of the triangle T_s of the triangulation of Ω, and appropriately Q_1, Q_2, Q_3 values defined by the formula (56). The function $r(\overline{x})$ is defined in the whole domain because triangles $\{\overline{T}_s\}_{s=1}^{n_e}$ cover it. The new mesh size function is defined as follows:

$$\gamma_{i+1}(\overline{x}) = \gamma_i(\overline{x})r(\overline{x}). \tag{58}$$

As $\mu \leq r(\overline{x}) \leq \lambda$ then $\mu\gamma_i(\overline{x}) \leq \gamma_{i+1}(\overline{x}) \leq \lambda\gamma_i(\overline{x})$.

It is easy to show that: $\exists \overline{x}, \overline{y} \in \overline{\Omega}$ such, that: $\mu\gamma_i(\overline{x}) = \gamma_{i+1}(\overline{x})$, and $\gamma_{i+1}(\overline{y}) = \lambda\gamma_i(\overline{y})$.
It can be shown, that

$$||\gamma_{i+1} - \gamma_i||_{\overline{\Omega},\max} \leq ||\gamma_i||_{\Omega,\max} \max\{|1-\mu|, |1-\lambda|\}, \tag{59}$$

where

$$||\gamma||_{\Omega,\max} := \max_{\overline{x}\in\Omega}\{|\gamma(\overline{x})|\}. \tag{60}$$

8. Numerical analysis of convergence

Many numerical simulations were performed by using the presented technique. In Fig. 3 an initial mesh for a subsonic inviscid potential flow is presented for the profile NACA0012 with angle of attack 5° with 0.1 Mach for the speed in the infinity. It is assumed, that speed has horizontal direction from left to right. Figure 4 shows the adapted mesh after three steps of remeshing.

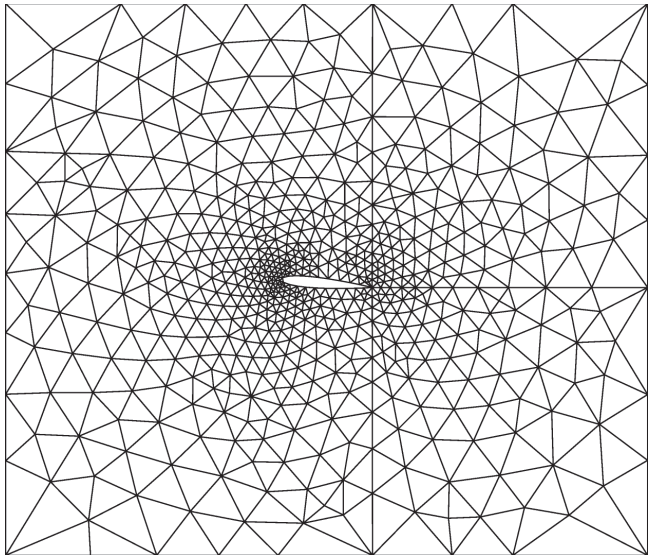

Fig. 3. Initial mesh for NACA0012 with angle of attack 5^0.

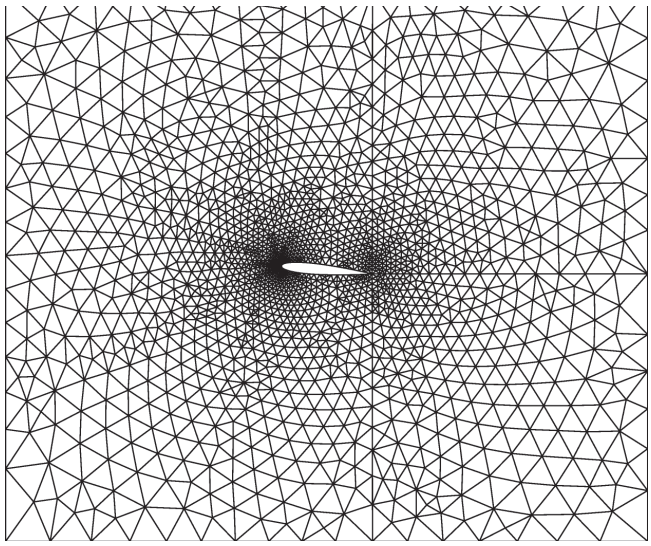

Fig. 4. Final mesh for NACA0012 with angle of attack 5^0.

Figure 5 contains the comparison of the values of the Mach number on the skin of the profile for the initial and final grid. It can be observed that the mesh rapidly becomes denser there where there is a bigger difference in the approximate solution. The presented final solution was compared with [11], so the final results obtained by the author were very close to those given in [11]. Figure 6 represents the graphs of the values of the Mach number on the skin of the profile for the initial, and the final grid for the profile NACA0012 with the angle of attack 1° with 0.7 Mach for the speed in the infinity and horizontal direction. That same situation like in the previous case can be observed – the adapted mesh becomes denser at the zones with greatest errors. In all cases initial meshes contained 600–900 elements, but thus final meshes 4000–6000 triangles.

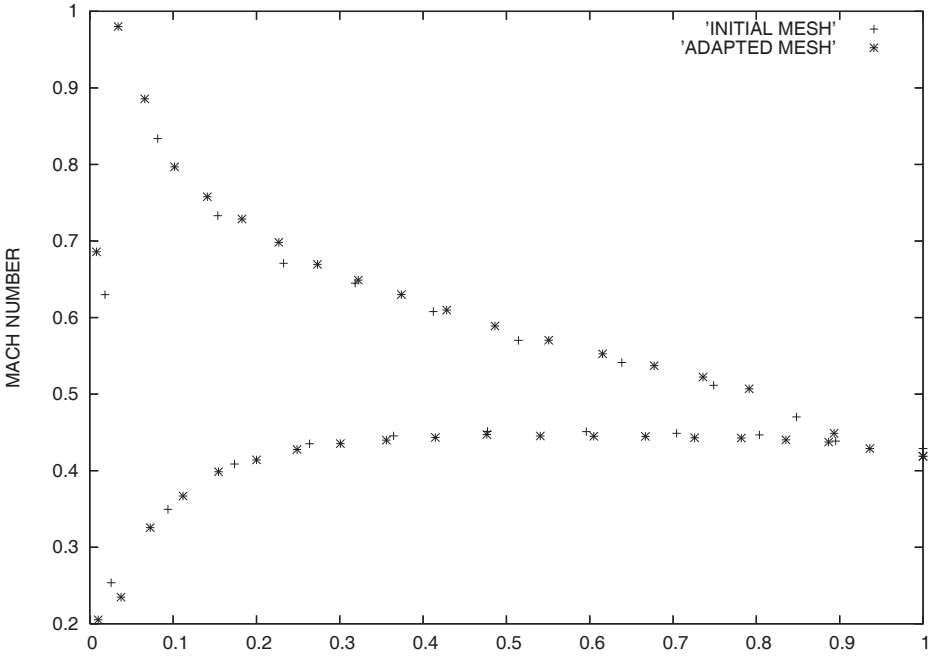

Fig. 5. Mach number on the skin corresponding to the initial and final grid for angle of attack 5°.

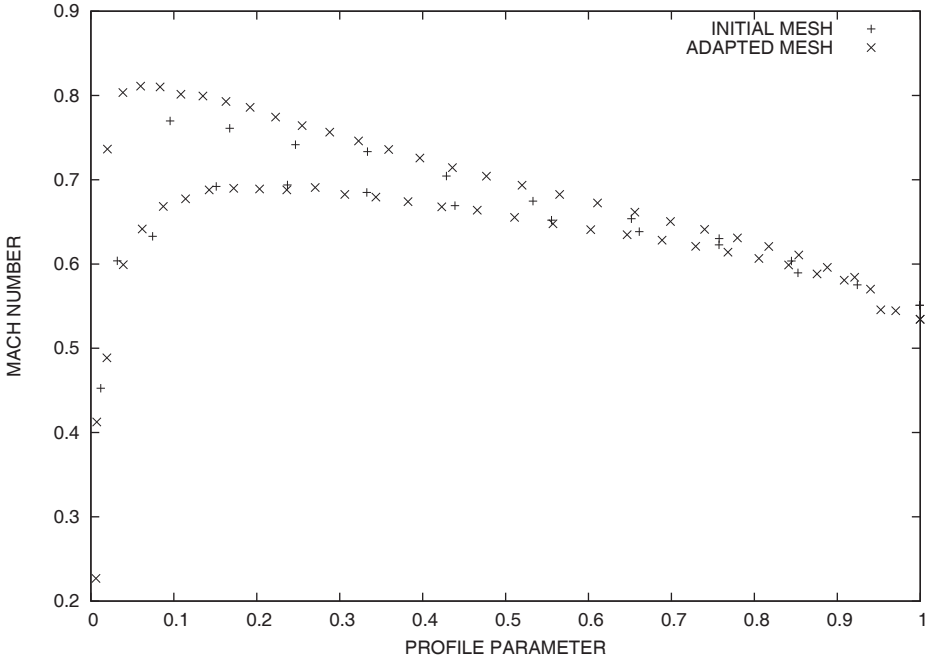

Fig. 6. Mach number on the skin corresponding to the initial and final grid for angle of attack 1°.

9. Conclusions

The developed generator of unstructured grids was applied to the algorithm of adaptation. The adaptation method was successfully applied to the problem of nonlinear subsonic flow. The convergence of the proposed algorithms was verified by the comparison of approximated solutions for two meshes on the boundary of the contact of gas and rigid body obstacle. The results were compared with other authors' [11]. It is concluded that the adapted mesh becomes denser in the zones of greater difference of those two solutions, what confirms the rightness of taken error indicators. The effectiveness of the proposed adaptation method depends on the grid generator, which takes into account the mesh size function. The combined advancing front method with Delaunay triangulation [1, 7] seems a good choice.

References

[1] H. Berger, G. Warnecke, W.L. Wendland. Analysis of a FEM/BEM transonic flow computations. *Universitaet Stuttgart, Mathematisches Institut A, Raport*, 1993.
[2] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland Publishing Company, Amsterdam, New York-Oxford, Studies in Mathematics and its Application, 1978.
[3] L. Demkowicz, J.T. Oden, W. Rachowicz, O. Hardy. Towards a universal h-p adaptive finite element strategy, Part 1: constrained approximation and data structure, *Comp. Meth. Appl. Mech. Engrg.*, **77**: 79–112, 1989.
[4] P. Dierckx. Curve and Surface Fitting with Splines. Oxford Science Publications, Clarendon Press, Oxford, 1995.
[5] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, J. Periaux. A distributed Lagrange multipliers/frictious domain method for the simulation of flow around moving rigid bodies: application to particular flow. *Comp. Meth. Appl. Mech. Engrg.*, **184**: 241–267, 2000.
[6] O.P. Jacquotte, J.L. Godard. Multidomain solution algorithm for potential flow simulation. *Comp. Meth. Appl. Mech. Engrg.*, No. 1–2, 1–19, 1994.
[7] J. Kucwaj. The Modelling and Triangulation of Geometrically Complicated Plane Domains and some Applications to Fluid Mechanics. Journal of Theoretical and Applied Mechanics, **35**: 2, 369–392, 1997.
[8] J. Kucwaj. The Algorithm of Adaptation by Using Graded Meshes Generator. *Computer Assisted Mechanics and Engineering Sciences*, **7**: 615–624, 2000.
[9] S.H. Lo. Finite element mesh generation and adaptive meshing. *Progress in Structural Engineering and Materials*, **4**: 381–399, 2002.
[10] J.T. Oden. h-p adaptive finite element methods for compressible and incompressible flows. *Computer Syst. in Engrg.*, **1**: 2–4, 523–534, 1990.
[11] A. Safjan, L. Demkowicz, D.P. Young. Compressible flow hp-Adaptivity and Kutta-Joukovsky condition, October 2006, The Boeing Company Project.
[12] Desheng Wang, Oubay Hassan, Kenneth Morgan, Nigel Weatherill. Enhanced remeshing from STL files with applications to surface grid generation. *Comm. in Num. Meth. in Engrg.*, **65**: 734–751, 2006.