
Journal of Engineering Mathematics **34:** 57–73, 1998.

Two-dimensional incompressible Navier-Stokes calculations in complex-shaped moving domains

KRIS RIEMSLAGH, JAN VIERENDEELS and ERIK DICK
Department of Mechanical and Thermal Engineering, University of Gent, B-9000 Gent, Belgium
e-mail: kris.riemslagh@rug.ac.be

Received 9 July 1997; accepted in final form 5 February 1998

Abstract. A method is described that allows the simulation of the flow of an incompressible fluid through complex-shaped two-dimensional domains which move in any prescribed time-varying way. The incompressible Navier-Stokes equations in arbitrary Lagrangian–Eulerian form are discretized on a triangular grid by means of a finite-volume method. Fully implicit time integration makes the method stable for any time step. Central differencing is used for the diffusive fluxes. Upwind differencing based on flux-difference splitting is used for the convective fluxes. A detailed description is provided for the discretization in two dimensions, with a collocated arrangement of pressure and velocity components as dependent variables. A description of the grid-generation process is given. Results are shown for the flow in a rotating-lobe pump.

Keywords: unstructured moving grids, arbitrary Lagrangian–Eulerian method (ALE), Navier-Stokes equations, incompressible flow, finite-volume method.

1. Introduction

The application we had in mind during the development of the method was the simulation of the flow through a lobe pump. A lobe pump is a rotative positive-displacement pump often used in the food industry, of which the behaviour is illustrated in Figure 1. Two externally driven rotors rotate contactless transporting fluid from suction to pressure side. The regions with cavitation and high shear rate are to be minimized or at least well controlled by proper design of the pump. At least a two-dimensional simulation of the pump is required to estimate pressure and shear-rate values accurately.

The complexity involved in an unsteady flow simulation increases if certain boundaries of the computational domain are allowed to move, so that the geometry of the domain changes with time. This means that the grid must be modified during the computation in order to accommodate these geometrical changes.

A first method which has proved to be successful for tackling such problems is the chimera approach, in which each individual geometry component has its own associated structured grid which moves independently of the other grids. Three-dimensional viscous simulations involving moving bodies have been produced by this method by Bunning *et al*. [1].

A second approach is the one used by Formaggia, Peraire and Morgan [2], applied to the solution of inviscid two-dimensional transient flows involving moving bodies. Their sequence of unstructured grids was generated by regridding parts of the flow field. At a particular time step the points on the moving boundaries are updated according to the movement. Then elements are deleted of which the shape and size differ significantly from the optimal value.

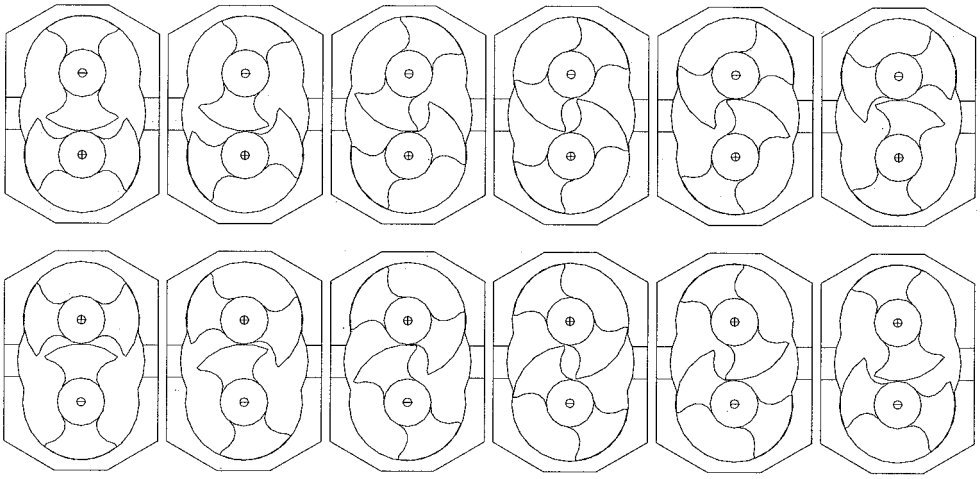

Figure 1. Two-dimensional illustration of behaviour of the lobe pump. Rotations per 15°.

These areas are then regridded. The solution on the new grid is determined by interpolation. The same approach is followed by Löhner [3].

A third technique in grid generation for moving boundaries uses mesh movement with a fixed grid connectivity. The prescribed velocity of the boundary nodes is the boundary condition of a problem of which the solution results in the velocity of every grid node. The operator that describes the nodal velocities can be as simple as a diffusion-like operator (Trépanier *et al*. [4]), ranging over a spring-analogon operator (Batina [5] and Palmerio [6]) towards a more complex pseudo-elasticity operator or a pseudo-pressure operator used by Palmerio [6]. Now that the connectivity is kept fixed at least during one time step, the conservation equations are expressed in the arbitrary Lagrangian–Eulerian way, so that they can be applied on control volumes which move with time. The approach with a pseudo-pressure operator is followed in the present work. Complications of the arbitrary Lagrangian–Eulerian formulation and the grid generation strategy are outlined.

2. Discretization

2.1. EQUATIONS

The equations describing the unsteady flow of an incompressible fluid are the Navier-Stokes equations. In the arbitrary Lagrangian–Eulerian formulation they are written for a moving control volume as follows:

$$\frac{\partial}{\partial t}\int_V \mathrm{d}V + \int_S (\underline{v} - \underline{v}_b)\cdot \underline{n}\,\mathrm{d}S = 0, \tag{1}$$

$$\frac{\partial}{\partial t}\int_V \rho\underline{v}\,\mathrm{d}V + \int_S (\rho\underline{v}(\underline{v} - \underline{v}_b) + p'\underline{\underline{I}} - \underline{\underline{\tau}})\cdot \underline{n}\,\mathrm{d}S = 0. \tag{2}$$

Here ρ is the density, $\underline{v}$ the velocity vector of the fluid in a fixed coordinate system, $\underline{v}_b$ the velocity vector of the boundary S of the control volume V, $\underline{n}$ the outer normal of this boundary, $\partial/\partial t$ the time derivative, p' the static pressure, $\underline{\underline{I}}$ the unit tensor, $\underline{\underline{\tau}}$ the viscous-stress tensor

defined for a Newtonian fluid by $\underline{\underline{\tau}} = 2\mu\underline{\underline{\gamma}}$ with $\underline{\underline{\gamma}}$ the rate-of-strain tensor $\gamma_{i,j} = \frac{1}{2}(\partial v_i/\partial x_j + \partial v_j/\partial x_i)$ and μ the dynamic viscosity.

In an Eulerian formulation $\underline{v}_b = 0$ and in a Lagrangian formulation $\underline{v}_b = \underline{v}$. In an arbitrary Lagrangian–Eulerian formulation the velocities $\underline{v}_b$ can be chosen independent of the flow. In a finite-volume discretization they are determined by the velocity of the control-volume edges.

Equation (1) expresses conservation of mass for a moving control volume and Equation (2) expresses conservation of momentum. If it is assumed that the dynamic viscosity is independent of the temperature, and that the fluid is incompressible, the equation for the conservation of energy can be solved separately.

The velocities $\underline{v}_b$ have to satisfy the space conservation law

$$\frac{\partial}{\partial t}\int_V \mathrm{d}V - \int_S \underline{v}_b \cdot \underline{n}\,\mathrm{d}S = 0. \tag{3}$$

Therefore Equation (1) can be written as

$$\int_S \underline{v} \cdot \underline{n}\,\mathrm{d}S = 0. \tag{4}$$

The Equations (3), (2) and (4) are discretized in a vertex-centred way by use of unstructured triangular grids. Control volumes are constructed by connecting midpoints of edges with centres of gravity of the surrounding triangles. Every node of the grid has a state vector

$$\mathbf{U} = \begin{pmatrix} u \\ v \\ p \end{pmatrix}, \tag{5}$$

where u and v are the Cartesian components of the velocity and $p = p'/\rho$ is the kinematic pressure.

Equation (4) is pre-multiplied with c^2, where c is a reference velocity. This pre-multiplication is necessary to ensure the same dimensions of the eigenvalues of the Jacobian matrix given in Section 2.3. Equation (4) becomes:

$$\int_S c^2 \underline{v} \cdot \underline{n}\,\mathrm{d}S = 0. \tag{6}$$

For the integration in time of (2) a reduced state vector is defined by $\mathbf{W} = J\,\mathbf{U}$, with

$$\mathbf{W} = \begin{pmatrix} u \\ v \\ 0 \end{pmatrix}, \qquad J = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{7}$$

Thus, Equation (2) can be written as

$$\frac{\partial}{\partial t}\int_V \mathbf{W}\,\mathrm{d}V + \int_S \underline{\mathbf{F}} \cdot \underline{n}\,\mathrm{d}S = 0, \tag{8}$$

where $\underline{\mathbf{F}}$ is the flux vector with contributions from the convective and the diffusive flux vectors.

Since the matrix J is singular, the set of Equations (8) cannot be solved by an explicit method.

2.2. TIME INTEGRATION

The time integration of (8)is done with the one-step implicit backward Euler method which is first-order accurate in time and A-stable. The state vector on the new time level $n+1$ is calculated from the state vector on the previous time levels n by

$$\frac{1}{\Delta t}\delta^n V\mathbf{W} + \int_S \underline{\mathbf{F}}^{n+1}\cdot\underline{n}^{n+1}\,\mathrm{d}S^{n+1} = 0, \tag{9}$$

where Δt is the time step and

$$\delta^n V\mathbf{W} = V^{n+1}\mathbf{W}^{n+1} - V^n\mathbf{W}^n. \tag{10}$$

The last term of Equation (9) can be written as

$$\int_S \underline{\mathbf{F}}^{n+1}\cdot\underline{n}^{n+1}\,\mathrm{d}S^{n+1} = \int_S \underline{\mathbf{F}}_c^{n+1}\cdot\underline{n}^{n+1}\,\mathrm{d}S^{n+1} - \int_S \underline{\mathbf{F}}_v^{n+1}\cdot\underline{n}^{n+1}\,\mathrm{d}S^{n+1}. \tag{11}$$

In this paper only first order in time is used. To increase the accuracy a second-order Crank-Nicolson scheme is suggested, since it does not increase the complexity of the method. In Equation (9) the change in time of $\mathbf{W}$ is assigned to the node of the control volume. This mass lumping also causes loss of accuracy.

2.3. SPACE DISCRETIZATION

The components $(\mathbf{f}_c, \mathbf{g}_c)$ of $\underline{\mathbf{F}}_c$, and $(\mathbf{f}_v, \mathbf{g}_v)$ of $\underline{\mathbf{F}}_v$ are

$$\mathbf{f}_c = \begin{pmatrix} u(u-u_b)+p \\ v(u-u_b) \\ c^2u \end{pmatrix}, \qquad \mathbf{g}_c = \begin{pmatrix} u(v-v_b) \\ v(v-v_b)+p \\ c^2v \end{pmatrix},$$
$$\mathbf{f}_v = \begin{pmatrix} \nu\dfrac{\partial u}{\partial x} \\ \nu\dfrac{\partial v}{\partial x} \\ 0 \end{pmatrix}, \qquad \mathbf{g}_v = \begin{pmatrix} \nu\dfrac{\partial u}{y} \\ \nu\dfrac{\partial v}{\partial y} \\ 0 \end{pmatrix}, \tag{12}$$

with $\nu = \mu/\rho$ the kinematic viscosity. We discretize the convective term of (11) using the polynomial flux difference splitting described in [7] and [8], but applied on an unstructured moving grid.

Differences of the convective fluxes can be written in algebraically exact form as follows

$$\Delta\underline{\mathbf{F}}_c\cdot\underline{n} = n_x\Delta\mathbf{f}_c + n_y\Delta\mathbf{g}_c = \begin{pmatrix} n_x\bar{u}+\bar{w}' & n_y\bar{u} & n_x \\ n_x\bar{v} & n_y\bar{v}+\bar{w}' & n_y \\ n_xc^2 & n_yc^2 & 0 \end{pmatrix}\Delta\begin{pmatrix} u \\ v \\ p \end{pmatrix}, \tag{13}$$

with $\bar{w}' = \bar{w} - w_b$, $\bar{w} = n_x\bar{u} + n_y\bar{v}$, $w_b = n_x u_b + n_y v_b$, where the bar means the algebraic mean of the differenced variables. The matrix is the discrete Jacobian, henceforth denoted by A.

For $n_x^2 + n_y^2 = 1$, the eigenvalues of A are

$$\lambda_1 = \bar{w}', \qquad \lambda_{2,3} = \tilde{w} \pm a, \tag{14}$$

with $\tilde{w} = (\bar{w} + \bar{w}')/2$ and $a = \sqrt{\tilde{w}^2 + c^2}$.

The corresponding left and right eigenvector matrices are given by

$$L = \begin{pmatrix} \dfrac{\bar{w}'\bar{v} + n_y c^2}{a'^2} & -\dfrac{\bar{w}'\bar{u} + n_x c^2}{a'^2} & \dfrac{n_x\bar{v} - n_y\bar{u}}{a'^2} \\ \dfrac{n_x}{2}\left(\dfrac{\tilde{w}}{a} + 1\right) & \dfrac{n_y}{2}\left(\dfrac{\tilde{w}}{a} + 1\right) & \dfrac{1}{2a} \\ \dfrac{n_x}{2}\left(\dfrac{\tilde{w}}{a} - 1\right) & \dfrac{n_y}{2}\left(\dfrac{\tilde{w}}{a} - 1\right) & \dfrac{1}{2a} \end{pmatrix},$$

$$R = \begin{pmatrix} n_y & \dfrac{\bar{u}}{a^+} - n_x\left(\dfrac{\bar{w}}{a^+} - 1\right) & \dfrac{\bar{u}}{a^-} - n_x\left(\dfrac{\bar{w}}{a^-} + 1\right) \\ -n_x & \dfrac{\bar{v}}{a^+} - n_y\left(\dfrac{\bar{w}}{a^+} - 1\right) & \dfrac{\bar{v}}{a^-} - n_y\left(\dfrac{\bar{w}}{a^-} + 1\right) \\ 0 & a^+ - \bar{w} & a^- + \bar{w} \end{pmatrix},$$

with $a^+ = a(1 + \delta)$, $a^- = a(1 - \delta)$, $a'^2 = \bar{w}\,\bar{w}' + c^2$ and $\delta = (\bar{w} - \bar{w}')/2a$.

The convective flux at the edge of the control volume between the nodes i and k is expressed in first-order form as

$$\mathbf{F}^1_{c,ik} = \tfrac{1}{2}(\mathbf{F}_{c,i} + \mathbf{F}_{c,k}) - \tfrac{1}{2}|A_{ik}|(\mathbf{U}_k - \mathbf{U}_i) = \mathbf{F}_{c,i} + A^-_{ik}(\mathbf{U}_k - \mathbf{U}_i) \tag{15}$$

with $|A| = R|\Lambda|L$, $|\Lambda| = \text{diag}\ (|\lambda_1|, |\lambda_2|, |\lambda_3|)$ and with $A^- = R\Lambda^- L$, $\Lambda^- = \text{diag}\ (\lambda_1^-, \lambda_2^-, \lambda_3^-)$, $\lambda^- = \min(0, \lambda)$.

The second-order discretization is constructed following a MUSCL-approach. The space derivatives that are needed, are computed with, in each triangle, the assumption that the state varies linearly. Limited space derivatives are then computed in each node from a limited combination of the space derivatives of the surrounding triangles. The minmod limiter is used. The left state at the boundary of the control volume is computed with these nodal limited space derivatives in the following way : $\mathbf{U}_l = \mathbf{U}_i + \underline{\mathbf{D}}_i \cdot \Delta\underline{x}$, with

$$\underline{\mathbf{D}}_i = \left(\frac{\partial \mathbf{U}_i}{\underline{x}}\right)_{\text{lim}}, \tag{16}$$

and $\Delta\underline{x}$ the vector pointing from node i to the centre of the edge connecting nodes i and k and lim denotes the limited combination of all gradients in the triangles surrounding node i. The right state is computed from node k in a similar way. In the MUSCL implementation Equation (15) is replaced by

$$\mathbf{F}^2_{c,ik} = \tfrac{1}{2}(\mathbf{F}_{c,l} + \mathbf{F}_{c,r}) - \tfrac{1}{2}|A_{lr}|(\mathbf{U}_r - \mathbf{U}_l). \tag{17}$$

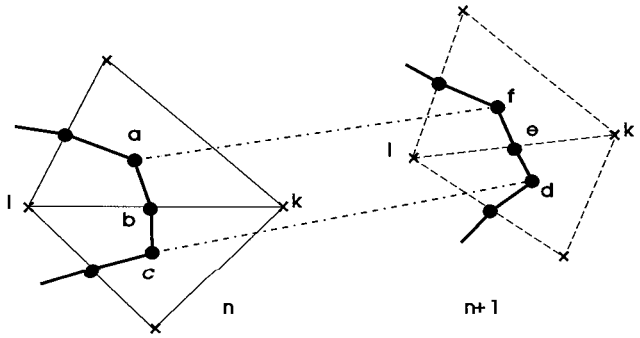

Figure 2. Movement of control-volume boundary during Δt

The above equation is rewritten in first-order form plus a second-order correction

$$\mathbf{F}^2_{c,ik} = \mathbf{F}^1_{c,ik} + Q_{c,ik}. \tag{18}$$

For the viscous term of (11) a central discretization is used which is exact for a linearly varying state [9]. Therefore, the complete discretization is second-order accurate in space.

2.4. SPACE CONSERVATION

When a time-integration scheme according to (9) is used, fluxes are computed at time level $n+1$. The discretized space conservation law (3) in node i takes the form

$$\delta V_i - \Delta t \sum_k \underline{v}_{b_{ik}} \cdot \underline{l}^{n+1}_{ik} = 0, \tag{19}$$

where the sum is taken over the surrounding nodes k and with $\delta V_i = V_i^{n+1} - V_i^n$. The subscript ik refers to the face of the control volume associated with the edge ik (ac and df in Figure 2). Since control volumes are formed by connecting midpoints of edges with centres of gravity of triangles (see Figure 2),

$$\underline{l}^{n+1}_{ik} = \underline{l}_{de} + \underline{l}_{ef} = \underline{l}_{df}, \tag{20}$$

where $\underline{l}$ is a vector in the direction normal to the boundary of the control volume and with size given by the length of the specific part of the boundary. From the basic geometric requirement follows that $\delta V_i = \sum_k \delta V_{ik}$ with V_{ik} as the volume swept by the part of the control-volume boundary between nodes i and k. This part is equal to $\text{Area}_{abcdefa}$ on Figure 2. For the computation of the convective fluxes only the normal component w_b of $\underline{v}_b$ must be known. This component w_b is calculated from

$$\underline{v}_{b_{ik}} \cdot \underline{l}_{df} \Delta t = w_{b_{ik}} \left| \underline{l}_{df} \right| \Delta t = \text{Area}_{abcdefa}, \tag{21}$$

and can be computed as soon as the grid at the new time level $n+1$ is known. With this choice of w_b, the space conservation law (3) is always satisfied.

2.5. BOUNDARY CONDITIONS

At solid boundaries the velocity vector is prescribed. This velocity vector is known, since the movement of the boundaries is given as a function of time. An equation must be constructed to compute the pressure. We consider two alternatives. First a characteristic combination of the equations can be taken [7], and second, the mass equation can be taken.

The characteristic combination is given by the second left eigenvector. For a solid non-moving boundary, this combination is given by

$$(n_x c, n_y c, 1). \tag{22}$$

This is a combination of the momentum equation in the direction normal to the wall and the mass equation. The viscous flux through the boundary edge in the normal momentum equation is given by

$$\int \nu \frac{\partial w}{\partial n} \, \mathrm{d}S, \tag{23}$$

where w stands for the outward normal component of the velocity. Using the mass equation in a coordinate system aligned with the boundary, we have

$$\frac{\partial w}{\partial n} + \frac{\partial v_s}{\partial s} + \frac{w}{R} = 0, \tag{24}$$

where v_s is the tangential velocity component and s is the tangential direction and R is the radius of curvature at the wall. Since obviously $\partial v_s/\partial s = 0$ and $w = 0$, we also have $\partial w/\partial n = 0$. This means that, for the viscous terms, there is no contribution from the boundary in the combination given by (22).

The second approach uses the mass equation to compute the pressure. In incompressible-flow calculations with a pressure-based iteration method, a pressure field, with a corresponding velocity field is computed, in such a way that, after each time step, the mass equation is satisfied. We can take this equation to determine the pressure. This approach seems unusual, in the sense that there is no pressure term in the original mass equation. However, due to the flux difference splitting, a pressure term arises.

At the inlet and outlet boundaries, the pressure is given. The velocities are computed from the momentum equations through a control-volume integration. The flux balance is closed with $\partial w/\partial n = 0$. If we also want to satisfy the mass equation at both the inlet and the outlet, we cannot drop the mass equation and prescribe the pressure instead. Therefore, the given pressure is used in the computation of the convective fluxes in the momentum equations. This means that flux-boundary conditions are used.

With this boundary condition, mass conservation is fulfilled in all control volumes. In regions where high pressure changes occur, the discretization error coming from the pressure terms in the mass equations can be large if the grid is not fine enough. This is a local error. This error is visible when the mass balance is verified through integration of the velocity field.

The use of a characteristic boundary condition results in a non-conservative scheme at the boundaries, which means that mass can disappear or enter through these boundaries. Therefore, we prefer the approach with the mass equation.

2.6. ITERATION METHOD

Having performed the space discretization, we can write the last term of Equation (9) for a node i as

$$\int_S \underline{\mathbf{F}} \cdot \underline{n}\, \mathrm{d}S = P_{i,0}\mathbf{U}_i + \sum_k P_{i,k}\mathbf{U}_k + Q_i, \tag{25}$$

$$Q_i = \sum_k Q_{c,ik}, \qquad P_{i,k} = A_{ik}^- + B_{ik}, \qquad P_{i,0} = -\sum_k P_{i,k}. \tag{26}$$

The sums are taken over all the neighbouring nodes k of node i. The coefficient matrices P have 3×3 components and contain contributions from the first-order convective discretization (A_{ik}^-) and contributions from the viscous discretization (B_{ik}). The term Q_i collects the second-order corrections.

Substitution of (25) in (9) yields the linear system,

$$\frac{1}{\Delta t} V_i^{n+1}\mathbf{W}_i^{n+1} + P_{i,0}^*\mathbf{U}_i^{n+1} + \sum_k P_{i,k}^*\mathbf{U}_k^{n+1} + Q_i^* = \frac{1}{\Delta t} V_i^n \mathbf{W}_i^n, \tag{27}$$

where P^* and Q^* are calculated with the state vector of the previous iteration level. The system (27) is solved for $\mathbf{U}^{n+1}$ with a standard ILU iteration.

The solution of (27) does not have to be determined more accurately than the truncation error of the time integration. 200 ILU iterations are carried out at every time step. This is sufficient to drop the residual 3 to 4 orders.

3. The mesh generation

3.1. DELAUNAY TRIANGULATION

The grid generation for an unsteady geometry with the complexity of a lobe pump is not easy. The use of unstructured grids is preferred. Every grid conforms to the position of the boundary at a prescribed time and spans the complete flow domain.

The discretization of the Navier-Stokes equations is done on an unstructured triangular mesh. The nodes are connected by a Delaunay triangulation algorithm which is performed in a locally scaled space. In a Delaunay triangulation nodes are connected such that no nodes of the triangulation lie inside the circumcircle of every triangle. When this is done in a locally scaled space, the circumcircle becomes an ellipse with dimensions that are determined by locally defined parameters.

3.2. MESH MOVEMENT

For every time step a grid is needed that remains valid. This means that no triangles are overlapping during the time step. During the time step the connectivity cannot be changed. In order to accommodate for large deformations the grid connectivity is allowed to change between two time steps. Also nodes can be added or removed. When nodes are added, a new state vector of flow variables has to be determined by interpolation. As is the case with node

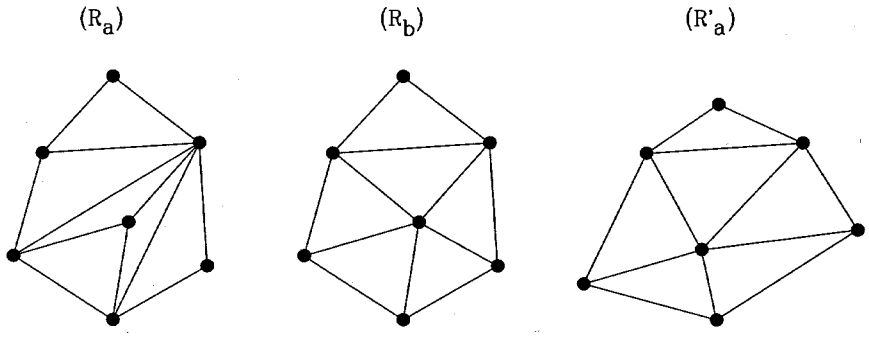

Figure 3. R_a: grid at time t; R_b: new grid at time t; R'_a: new grid at time $t + \Delta t$.

removal, the state vector of neighbouring nodes can be changed to maintain conservation of quantities.

The problem can be formulated as follows. Given the solution of the flow problem at a certain moment in time, t, on a grid R_a, find a new grid R_b, also at time t, and a new grid R'_a at time $t + \Delta t$, where R_b and R'_a have the same connectivity, and R_a and R_b have as many nodes in common as possible. This is illustrated in Figure 3.

At every time step the grid R_b is constructed by modification and adaptation of the grid R_a. First, the grid is made Delaulay in a localy scaled space by an edge-swapping algorithm. Doing this, we do not change the position of the nodes; only the connectivity is changed. Then, adaptation based on locally specified mesh spacings is done by addition and removal of nodes.

Since R_a and R_b are both at the same time level, the solution can be interpolated safely from R_a to R_b. To calculate the solution from R_b to R'_a, the Navier-Stokes equations need to be integrated.

3.3. Grid-generation tools

To construct the grid R'_a from the grid R_b, nodes can be moved, but the connectivity must be kept fixed. The boundaries change from R_b to R'_a. The position of the internal nodes of R'_a must be determined in connection with the condition that this grid is valid. This is called the moving-grid problem. To generate a valid grid, a system of equations is constructed expressing the equilibrium of a set of forces acting on the nodes. The forces that act on a node depend on the position of that node and the surrounding nodes. The solution of the set of equations is the position of every grid node.

For the forces acting on node a the equilibrium can be written as

$$\sum F_{a,no} = 0, \tag{28}$$

with the sum taken over all the neighbouring triangles of node a.

Defining $\zeta = h_a/s_a$ for a node a of a triangle abc, with h_a the height of the triangle and s_a the length of the edge bc, we define the force $F_{a,no}$ by

$$F_{a,no} = \frac{1}{\zeta}\overline{e}_s \quad \text{if} \quad \zeta > \zeta_0, \qquad F_{a,no} = \left(\frac{2}{\zeta_0} - \frac{\zeta}{\zeta_0^2}\right)\overline{e}_s \quad \text{if} \quad \zeta \leqslant \zeta_0, \tag{29}$$

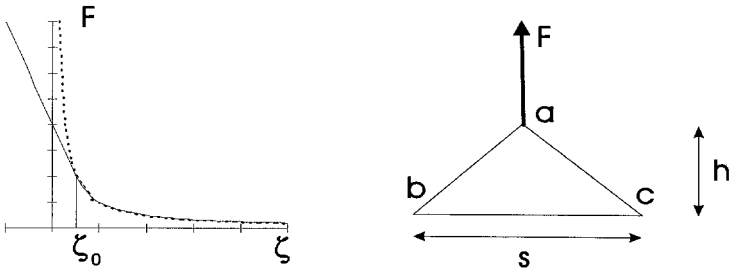

Figure 4. Force of triangle *abc* on node *a*.

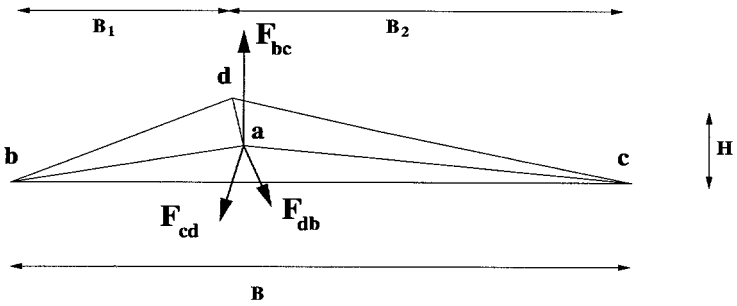

Figure 5. ζ_0 must be chosen such that node *a* remains inside its surrounding polygon: bcd.

as is illustrated in Figure 4, with $\overline{e}_s$ being the unit vector perpendicular to the edge *bc* in the direction of node *a*. The signed height is calculated as

$$h_a = \frac{2A}{s_a}, \qquad A = \sum \frac{x_i + x_{i+1}}{2}(y_{i+1} - y_i), \tag{30}$$

where A is the signed area of the triangle and the sum is positive if the edges of the triangle are visited counter-clockwise. For a valid triangulation all areas are positive.

In the construction process of R'_a from R_b, after the boundary nodes have been given their new position, the internal nodes are not located in the equilibrium position. Possibly a number of triangles have changed orientation, so a number of edges are intersecting and some triangles have a negative area. Starting from the old location of the internal nodes, we perform an iteration process on the set of Equations (28), moving the nodes towards their final positions.

The parameter ζ_0 is introduced only to allow negative values of ζ during the solution process. The solution of the grid movement should be independent of ζ_0: *i.e.* all $\zeta > \zeta_0$. This will be the case if ζ_0 is chosen small enough.

3.4. CHOICE OF ζ_0

Consider the surrounding polygon *bcd* around node *a* (Figure 5). The forces that act on node *a* are F_{bc}, F_{cd} and F_{db}. Triangle *abc* will remain positive if node *a* is located within the polygon *bcd*. When the node *a* is moved across the edge *bc*, it will be forced to return inside the polygon when

$$|F_{bc}| > |F_{cd}| + |F_{db}|. \tag{31}$$

This is a sufficient condition to keep the triangle positive. When node *a* is located on the edge *bc*, then $\zeta = 0$ for triangle *abc*, so according to Equation (29)

$$|F_{bc}| = 2/\zeta_0. \tag{32}$$

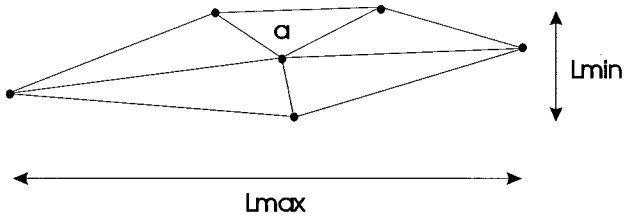

Figure 6. Polygon surrounding node a with indication of values $L_{\max}$ and $L_{\min}$.

In this case the forces F_{cd} and F_{db} are

$$|F_{cd}| = \frac{B_2}{H}, \qquad |F_{db}| = \frac{B_1}{H}. \tag{33}$$

Combining the previous equations, we find that the triangle abc will remain positive if

$$\frac{2}{\zeta_0} > \frac{B_1 + B_2}{H} = \frac{B}{H}. \tag{34}$$

We might repeat the discussion for the surrounding polygons that have more than three nodes, but the result will be identical. In most cases the condition (34) is far too stringent.

From this we conclude that at equilibrium all triangles are positive if ζ_0 is chosen such that $\zeta_0 < 2L_{\min}/L_{\max}$ with $L_{\min}$ and $L_{\max}$ defined as the minimum and maximum dimensions of the smallest rectangle surrounding the polygon (see Figure 6). A typical value of ζ_0 is 0·01.

3.5. Nested Iterations

Because the grid has to be rotated during every time step, an efficient solver is required to find the solution of (28). Nested iteration with coarser meshes is used to solve the problem. First, the moving-grid problem is solved on the coarsest mesh. When all the triangles have a positive surface area, the problem is solved on the next finer mesh starting with interpolated values for the displacements on the the coarser mesh. At every level the solution of the set of Equations (28) is solved by a Jacobi iteration after a Newton linearization. The linearization is recalculated at every Jacobi iteration. Typically, 20 or less Jacobi iterations will solve the problem.

3.6. Mesh coarsening

As explained in the previous section, nested iteration is used to speed up the calculation of the new position of the grid nodes. The coarser meshes are generated automatically as described in [10].

The coarser meshes have so-called telescoping nodes. This means that all nodes of a coarse mesh appear as nodes in all the finer meshes. The meshes are generated from fine to coarse. To generate a coarser mesh for a given fine mesh, the following is done:

– First, a set of nodes from the fine mesh is selected to appear in the coarser mesh. A selection algorithm described by H. Guillard [11] is used. We flag nodes for removal by looping over all the nodes and doing the following for every node. If the node is not flagged, all its neighbours in the grid are flagged; if the node is flagged, nothing is done;

– Then this set of nodes is connected to form a triangulation. Starting from the fine mesh, we remove flagged nodes one by one. The polygonal cavity that remains after removal of a node and all its connecting edges is retriangulated by a Delaunay algorithm.

The fact that the nodes of the coarse mesh appear in the finer mesh simplyfies the transfer of data between the meshes.

4. Evolution of the flow calculation

The flow solver coupled with grid-generation results in the following procedure:

(1) Define geometry and determine movement of the boundary;
(2) Generate starting grid R^0 that can be used for the flow calculation ($R_a^0 = R_b^0 = R^0$);
(3) Generate, with R_a^0 as the finest grid, coarser grids that are needed for the nested iteration;
(4) Determine initial state vectors for the flow calculation, $\mathbf{U}^0 = (0, 0, p_0)^T$, with $p_0 = p_{\text{inlet}}$;
(5) Determine an appropriate Δt. This Δt is constant during the calculation. Here Δt is chosen such that the rotors rotate 1°. Since an implicit time integration method is used, there is no time-step limitation for stability. The limitation comes from the fact that the construction of the grid has to be possible. With the same connectivity there must exist a distribution of the nodes, so that all triangles are valid, while the boundaries are rotated. And it must be possible to calculate the position of the nodes;
(6) Generate the grid for the next time step R_a^{n+1}. This grid must have the same connectivity as R_b^n;

– Generate the coarsest grid first. Move the boundary nodes and solve the set of Equations (28) for the internal nodes;
– Interpolate the position on the next finer mesh. Then, on this mesh solve, the set of Equations (28);

Repeat this until the finest mesh;

(7) Determine the normal velocity component of the speed of the boundary of the control volumes from Equation (21), satisfying space conservation;
(8) Find the solution of the flow equations for the new time step. This is done by solving system (27) for all nodes. The solution at the previous time level is used as the initial solution to start from. 200 ILU iterations with an underrelaxation $\omega = 0{\cdot}75$ are done to calculate the solution. The linearization of the system is recalculated every 5 iterations;
(9) Generate grid R_b^{n+1} by modifying and adapting grid R_a^{n+1}. First, the grid is made Delaulay in a localy scaled space. Doing this, the position of the nodes is not changed, only the connectivity is. Then, adaptation based on locally specified mesh spacings is done by addition and removal of nodes. For the nodes that are added flow variables are interpolated. Because the connectivity is changed, the definition of the control volumes is changed and the values V^{n+1} have to be recalculated. V^{n+1} is equal to V^n during the next time step;
(10) If the number of time steps is less than the required number, n is incremented by one. Then go to 6.

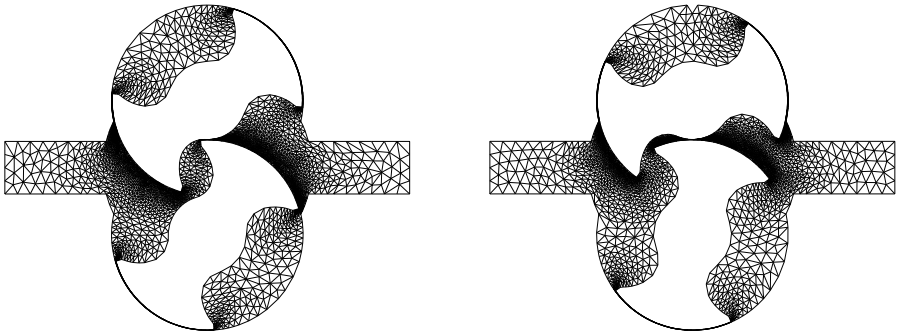

Figure 7. Grids used for the flow calculation on the winglobe pump at stages 55° (left) and 75° (right).

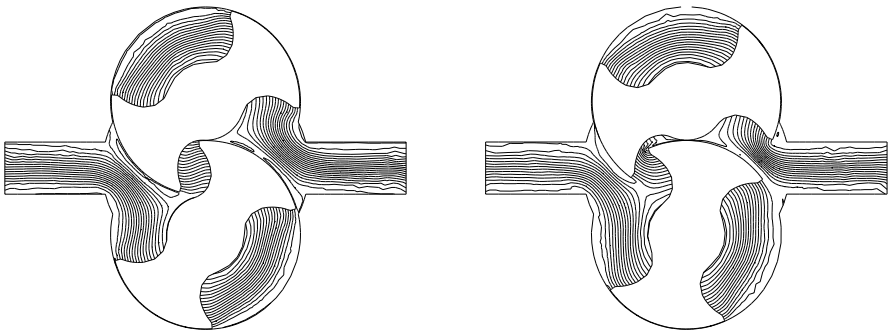

Figure 8. Result of flow calculation by means of streamlines at stages 55° and 75°.

5. Results

The flow through a winglobe pump is simulated (Figure 7). The rotor diameter is 131 mm. The speed of rotation is 360 rpm. The kinematic viscosity of the fluid is $\nu = 7{\cdot}6$ mm^2/s, the density is 1000 kg/m^3. As to the boundary conditions, pressure is prescribed both at the inlet (1 bar) and the outlet (2 bar), resulting in a pressure rise over the pump of 1 bar. The absolute value of the pressure is of no importance in incompressible flow simulations. The flow calculation is done with 1° increments per time step. Grids are calculated in a sequence where the connectivity is kept fixed during 1° of rotor rotation.

As can be seen in Figure 7, the nodes are clustered near some regions of the rotors to assure sufficient node density in the clearences between the rotors and the housing and between the two rotors themselves. This was achieved through the use of local mesh-spacing parameters. Two grids are shown with clockwise top-rotor rotations of 55° and 75°, respectively.

The number of nodes of a grid is approximately 8500. Since nodes are added and removed at every time step, the exact number changes during the calculation. For such a grid 200 ILU iterations take 18 minutes on a HP workstation HP9000-715/100XC, while the grid generation and movement needs about 1 minute for every time step. We do not use the most efficient implementation here. The authors believe that the execution time can be reduced by a factor of up to 20 by optimizing the implementation.

In Figure 8 the evolution of the flow calculation is illustrated by means of streamlines. As can be seen, the streamlines are calculated in an absolute frame. Therefore they enter the rotors. One can see that there is a vortex present near the end of the rotors where the flow is in the opposite direction.

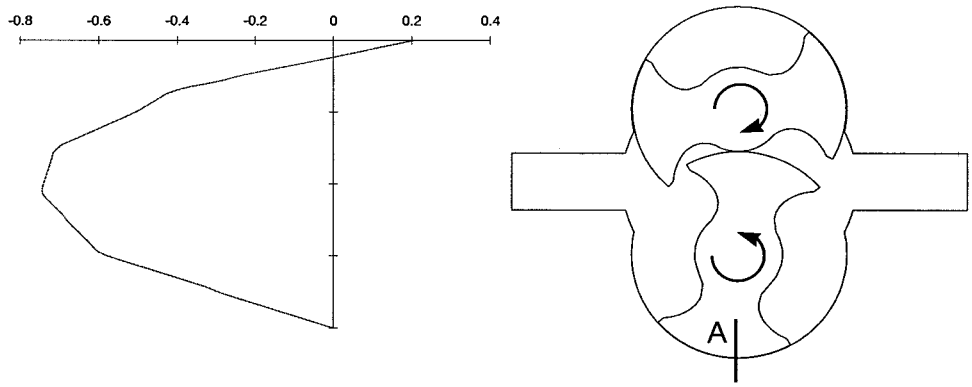

Figure 9. Velocity profile (m/s) in clearance between stator and rotor at location A.

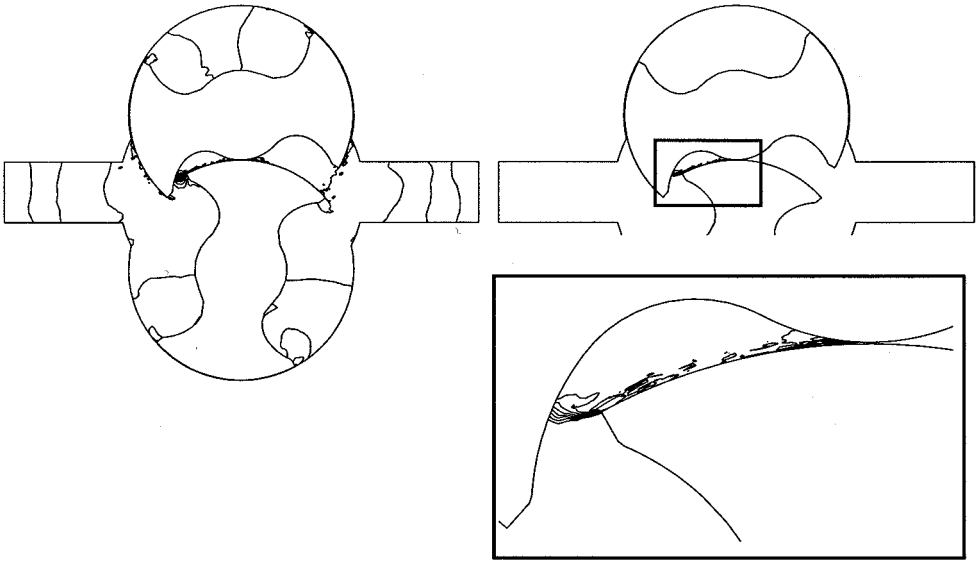

Figure 10. Pressure contours per 0·5 m^2/s^2. Left: between minimum and maximum pressure. Right: between 85 m^2/s^2 and 95 m^2/s^2.

In Figure 9 the velocity profile is given in the clearance between the rotor and the stator at a location A (see figure). The profile corresponds with a driven couette flow. One can see that this is a flow with high shear values, and also that there is a leak flow present. The actual values can be easily computed from the numerical results.

Figure 10 shows the pressure contours at the same rotor position. The kinematic pressure at the inlet is 100 m^2/s^2. Going inside the lobe pump the pressure further drops to a minimum of 85 m^2/s^2. Over the clearences there is a pressure rise. The maximum of the pressure is 203·6 m^2/s^2. At the right of the figure the lowest pressure contour lines are given. In real-life situations cavitation bubbles can appear at these locations, if the inlet pressure of the lobe pump is low enough. Although no cavitation model is included in the simulation flow, situations can be investigated that would lead to cavitation.

No comparison can be made with experimental results. Experimental data are not available for the test cases illustrated in this paper.

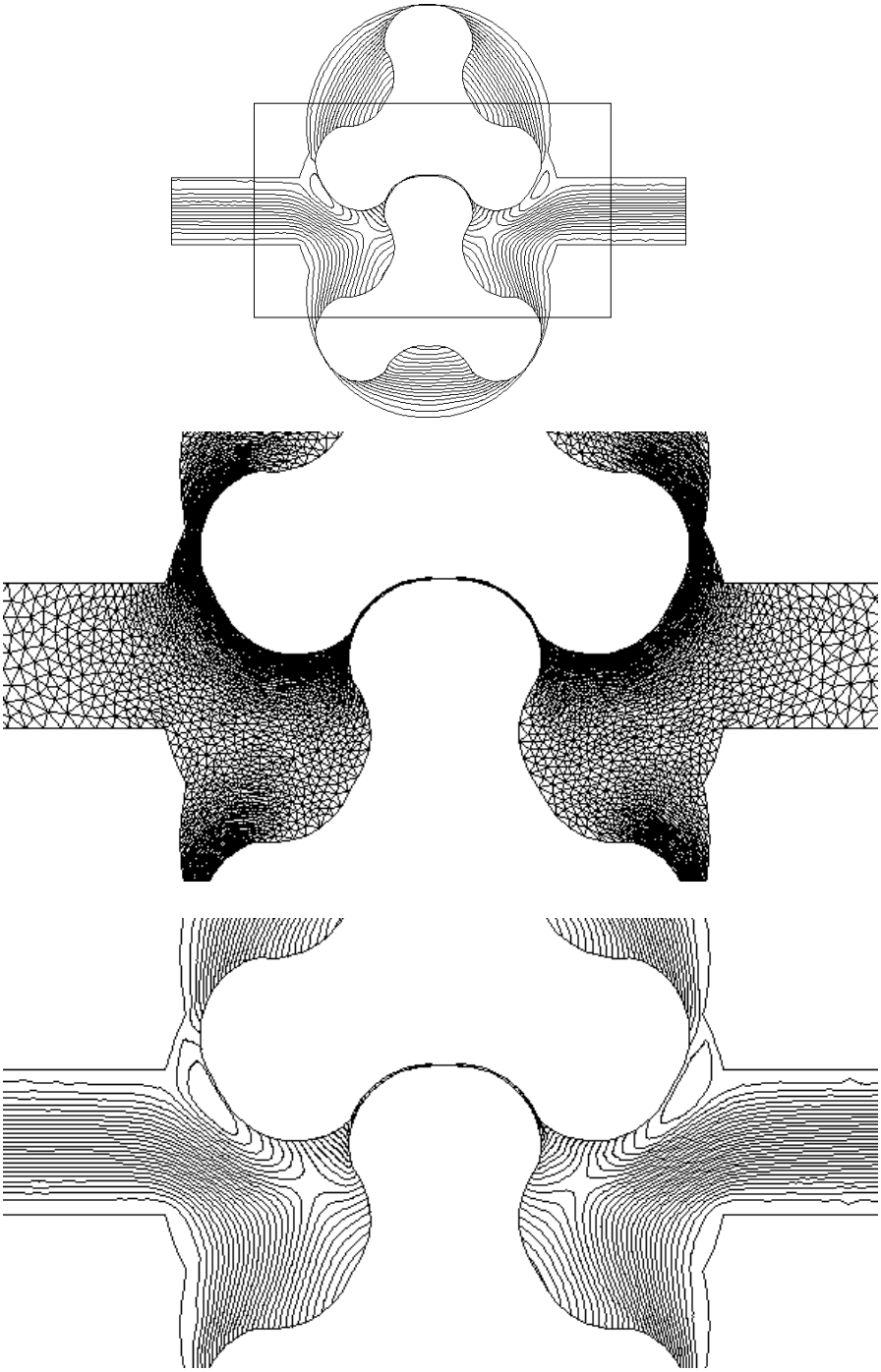

Figure 11. Grid and streamlines of tri-lobe pump at stage 0°.

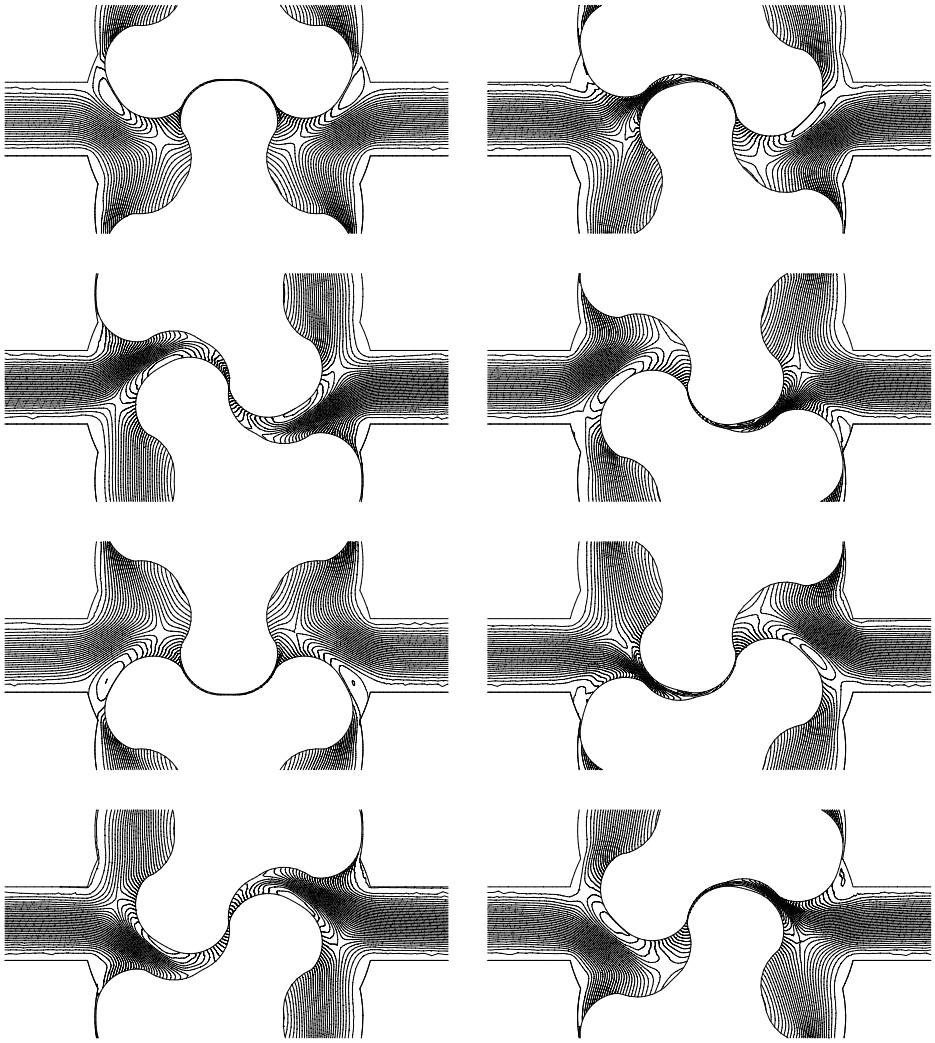

Figure 12. Evolution of flow calculation by means of streamlines shown for rotor positions from 0° to 105° with steps of 15°.

A second simulation was performed on a tri-lobe pump. The rotor diameter is 132·5 mm. The speed of rotation is 460 rpm. The kinematic viscosity of the fluid is $\nu = 46$ mm^2/s, the density is 880 kg/m^3. The pressure rise over the pump is 2 bar. Figure 11 shows grid and streamlines with the top-rotor position at stage 0°. In Figure 12 the evolution of the flow calculation is illustrated by means of streamlines.

6. Conclusion

In the paper a procedure is proposed for two-dimensional incompressible-flow calculations in complex-moving domains. The procedure includes the mesh-movement strategy, as well as the discretization scheme.

The algorithm is illustrated on the complex geometry of rotating-lobe pumps. Results of two different simulations are shown. Although no comparison could be made with experimental data, it is clearly shown that such simulations give important flow information to the designer of this kind of pump. Both high-shear-rate areas, that could cause degradation of some fluids, as well as low pressure areas that initiate cavitation, can be easely identified. The current simulation model can therefore be considered as an important design tool for this kind of positive-displacement pumps.

Acknowledgements

The research reported here was granted by contract IWT/OZM/94005 of the Flemish Institute for the Stimulation of Scientific-Technological Research in Industry (IWT), and by contract GOA-95003 of the concerted action program of the university of Gent supported by the Flemish Government.

References

1. P. G. Bunning, I. T. Chiu, S. Obayashi, Y. M. Rizk and J. L. Steger, *Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent.* AIAA-88-4359-CP (1988) pp. 1–6.
2. L. Formaggia, J. Peraire and K. Morgan, Simulation of a store separation using the finite-element method. *Appl. Math. Mod.* 12 (1988) 175–181.
3. R. Löhner, An adaptive finite-element solver for transient problems with moving bodies. *Comp. Struct.* 30 (1988) 303–317.
4. J. Y. Trépanier, M. Reggio, M. Paraschivoiu and R. Camarero, Unsteady Euler solutions for arbitrarily moving bodies and boundaries. AIAA *J.* 31 (1993) 1869–1876.
5. J. T. Batina, *Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes*. AIAA-89-0115 (1989) pp. 1–10.
6. B. Palmerio, Coupling mesh and flow in viscous fluid calculations when using unstructured triangular finite-elements. *Int. J. Comp. Fluid Dyn.* 6 (1996) 275–290.
7. E. Dick, Multigrid methods for steady Euler and Navier-Stokes equations based on polynomial flux-difference splitting. In: W. Hachbush and U. Trottenberg (eds.), *Proc. of the Third European Conference on Multigrid Methods, Bonn, 1991.* Birkhauser Int. Series on Numerical Mathematics 98 (1991) pp. 1–20.
8. E. Dick and J. Linden, A multigrid method for steady incompressible Navier-Stokes equations based on flux difference splitting. *Int. J. Num. Methods for Fluids* 14 (1992) 1311–1323.
9. T. J. Barth, Aspects on unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations. In: H. Deconinck and T. Barth (eds.), *AGARD Report 787, VKI Special Course on Unstructured Grid Methods for Advection Dominated Flows* (1992) pp. 6.1–6.61.
10. K. Riemslagh, Unstructured multigrid: formulation, mesh generation and datastructure. In: N. P. Weatherill, P. R. Eiseman, J. Hauser and J. F. Thompson (eds.), *Proc. of the 4th International Conference on Numerical Grid Generation in CFD and related Fields, Swansea, 1994.* Pineridge Press (1994) pp. 199–210.
11. H. Guillard, Node-nested multigrid with Delaunay coarsening. In: *INRIA, Rapport de recherche*, 1898 (1993) pp. 1–16.