TASCOPT - A Shape Optimization Plateform for Turbomachinery Applications

Mugurel Stanciu, Bijan Mohammadi

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

TASCOPT - *A Shape Optimization Plateform for Turbomachinery Applications*

Mugurel Stanciu Bijan Mohammadi

No 3803

November 1999

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--3803--FR+ENG

INRIA
ROCQUENCOURT

TASCOPT - A Shape Optimization Plateform for Turbomachinery Applications

Mugurel Stanciu * Bijan Mohammadi †

Thème 4 — Simulation et optimisation
de systèmes complexes

Projet M3N

Rapport de recherche n ° 3803 — November 1999 — 69 pages

Abstract: We present *TASOPT*, a new plateform for shape optimization in rotating frames for in particular applications in turbomachinery. The theoretical background comes from the idea develloped at INRIA for aerodynamical shape optimization.

TASOPT adopts the unstructured CAD-free framework for shape and mesh deformations where the interface with CAD is realized at the final stage of the optimization. This framework is extended to 2D and 3D multi-block structured grids. Automatic differentiation through *Odyssée* is used to access discrete sensitivities in reverse mode. In addition, the incomplete gradient approach for functionals based on boundary integrals enables for a drastic reduction of the computational cost. The plateform uses various flow solvers depending on the physic of the problem ranging from incompressible to high speed flows. Various minimization algorithm based on linear and quadratic models for the functional are presented.

These ingredients are illustrated through a first cascade configuration blade shape optimization. In addition, we present an exemple of the application of the plateform in industrial environment for VALEO-TM in the design of 3D cooling engine devices. In particular, we show how the plateform has been interfaced with the commercial flow solver *TASCFLOW*. This extension requires the developpement of specific interface between *TASCOPT*, *TASCFLOW* and CAD tools used by VALEO.

Key-words: Aerodynamic Shape Design, Gradient Methods, Automatic Differentiation, Incomplete sensitivities, CAD-free Framework, Turbomachinery, Multi-block Structured Grids

(Résumé : tsvp)

* e-mail: Mugurel.Stanciu@inria.fr
† e-mail: Bijan.Mohammadi@inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

TASCOPT - une Plateforme d'Optimisation de Formes pour des Applications Turbomachines

Résumé : Nous présentons la plateforme d'optimisation de formes TASCOPT pour les configurations en repères tournants et en particulier les applications turbomachines. Cet outil utilise les résultats de recherche amont entrepris à l'INRIA depuis de nombreuses années.

TASCOPT reprend l'environnement CAD-free pour la gestion des déformations de la forme et du maillage sans faire appel à une parametrisation CAO pendant l'optimisation, tout en permettant un retour sur la CAO en finale. Cette approche est généralisée pour des maillages structurés multi-bloc et non-structurés 2D et 3D. La différentiation automatique par *Odyssée* est utilisé pour

le calcul des sensibilités en mode inverse. Par ailleurs, une approximation des gradients dans le cas des fonctionnelles integrales de bord permet une réduction de coût de calcul significative. La plateforme utilise divers solveur fluide suivant la physique du problème allant de l'incompressible au écoulements à grande vitesse. Plusieurs méthodes de minimisation basées sur des approximations linéaires et quadratiques de la fonctionnelle coût sont utilisées.

Ces points sont illustrés sur une configuration de cascade de pale de ventilateur où notamment le choix du critère d'optimisation est discuté. Par ailleurs, nous présentons un exemple d'application de la plateforme en environnement industriel pour 'VALEO Thermique Moteur' en conception de systèmes de refroidissement moteur, ainsi que l'adaptation de la plateforme au logiciel industriel *TASCFLOW*. Cette application a exigé le développement d'interfaces spécifiques entre *TASCOPT*, *TASCFLOW* et les outils CAO utilisés chez VALEO.

Mots-clé : Optimisation de Formes Aérodynamique, Méthode de Gradient, Différentiation Automatique, Gradient Incomplet, Paramétrisation CAD-Free, Turbomachines, Maillages Multi-blocs Structurés.

Contents

1 Shape Optimization for Steady Flows **5**

1.1 Introduction 5

1.2 Optimization Problem 6

1.3 Framework 6

1.3.1 Cost function definition 6

1.3.2 Geometrical constraints 7

1.3.3 State constraints 7

1.3.4 Smoothing operator 9

1.3.5 Mesh deformation 10

1.3.6 Optimization algorithm 12

1.4 Gradient computation 12

1.4.1 Automatic differentiation 12

1.4.2 Incomplete sensitivities 13

1.5 Optimization methods 13

1.5.1 Steepest descent method 13

1.5.2 Heavy-ball method 14

1.5.3 BFGS method 14

1.6 Turbomachinery Flows 15

1.6.1 Blade cascade 15

1.6.2 2D case 16

1.6.3 Fan efficiency 17

1.6.4 Fan optimization criteria 17

2 TASCflow Environment **19**

2.1 General description 19

2.2 TASCflow capabilities 20

2.2.1 Governing equations 20

2.2.2 Discretization 24

2.2.3 Boundary conditions 25

2.3 Different meshes used in TASCOPT-3.0 29

2.3.1 Mono-block grids 30

2.3.2 Multi-block grids with one block around the profile 37

2.3.3 Multi-block grids with several blocks around the profile 40

3 Optimization Code TASCOPT-3.0 **46**

3.1 Code description 46

3.1.1 Previous versions 46

3.1.2 Actual version 46

3.1.3 Shell OPTIMTASC 47

3.1.4 Optimization code optimtasc.exe 49

3.2 Input and output files 51

3.2.1 Input file 51

3.2.2 Output files 55

4 Numerical Results **57**

4.1 NACA 65-010 57

4.2 FC1001 62

5 CONCLUSIONS **67**

References **68**

Chapter 1

Shape Optimization for Steady Flows

1.1 Introduction

Nowadays, shape optimization in aerodynamics is still not used on a large scale in industrial environments, due to multiple mathematical and computational problems. The optimization algorithm presented here is based on a feasible, acceptably accurate approach for systematic industrial application.

We developed a new code for shape optimization in turbomachinery. TASCOPT was the result of the collaboration VALEO TM - INRIA Rocquencourt and consisted of the application of the theoretical researches on optimization obtained and validated at INRIA in an industrial environment, for the specific problems of VALEO TM. This code is used for the optimization of the engine cooling fan system, and its purpose is to obtain new cooling fans with better efficiency and less noise.

We used the approach for shape optimization in aerodynamics problems developed by Prof. Mohammadi. We present in this report the main ingredients of this approach. The unstructured CAD-free framework for shape and mesh deformations was extended the a CAD-free framework to multi-block structured grids and its techniques were coupled with the commercial flow solver TASCflow. Specific interfaces were developed between the optimization code and the flow solver in order treat the CAD descriptions used at VALEO TM. The automatic differentiation of programs for computing the discrete adjoint operator and the gradient approximation based on local informations on the shape were implemented in the code. Due to these approximations, the gradient computation which becomes fast enough to allow the possibility of complex shape optimizations. Several optimization gradient methods based on linear and quadratic approximations of the cost function are used. The optimization algorithm is based on a progressive optimization technique, meaning that every intermediate solution is not fully converged, in order to reduce the computational time. This approach was widely tested in in-viscid incompressible and compressible flow optimization problems and it proved to be a powerful tool which can treat realistic cases and can lead to future systematic industrial applications.

We present here the mathematical formulation, the optimization algorithm, and its applications for the study of a 2D fan blade profile: the choice of optimization criteria and the computational procedure.

The presented 2D results serve as a basis for the further optimization of 3D blades. Future improvements of TASCOPT are planned. New mesh deformation techniques using hyperbolic operators and a BFGS method using incomplete Hessians are under study. The limited 3D capabilities of TASCOPT will be extended to full 3D description of fan blades.

1.2 Optimization Problem

We consider the following problem:

$$\begin{cases} \min_{x_c} j(x_c, U(x_c), \nabla_{x_c} U(x_c)) \\ E(x_c, U(x_c), \nabla_{x_c} U(x_c)) = 0 \\ g_1(x_c) \leq 0 \\ g_2(U(x_c)) \leq 0 \end{cases} \tag{1.1}$$

where x_c are the control points (the blade shape), U the flow variables (velocity, pressure, turbulent energy, etc.), E the state equations (Reynolds-averaged Navier-Stokes equations and a turbulence model), g_1 the geometrical constraints, g_2 the state constraints and j the cost function subject to minimization.

1.3 Framework

We have chosen that in our case the control variables correspond to the mesh wall nodes (CAD-free approach): $x_c \equiv x_w$. The only geometrical entity available during the optimization is the mesh. As a gradient method is used in the optimization procedure, we need to compute the gradient of the cost and the constraints with respect to the control variables x_c ($dj/dx_c, dg_1/dx_c$, dg_2/dx_c). By taking into account the constraints (e.g. lift, volume) as penalty terms, a new cost function J is defined as a weighted linear combination of the original cost function j and the constraints g_1 and g_2.

1.3.1 Cost function definition

Since the gradient approximation is based on the flow solution on the shape and since the computed gradient is not sensitive to the complete domain, the cost function should depend only on the informations on the profile.

Usually the optimization criteria are based on the aerodynamic coefficients, i.e. on the boundary integrals calculated on the profile. The aerodynamic coefficients C_d, C_l and C_m are:

$$
\begin{aligned}
C_l &= \frac{\int_\Gamma [T \cdot n]_n \, d\sigma}{\frac{1}{2}\rho_\infty |u_\infty|^2 A} \\
C_d &= \frac{\int_{\Gamma_w} [T \cdot n]_t \, d\sigma}{\frac{1}{2}\rho_\infty |u_\infty|^2 A} \\
C_m &= \frac{\int_{\Gamma_w} (x_n [T \cdot n]_t + x_t [T \cdot n]_n) \, d\sigma}{\frac{1}{2}\rho_\infty |u_\infty|^2 A c}
\end{aligned}
\tag{1.2}
$$

with $T = p\,I - (\nu + \nu_t)(\nabla u + \nabla u^T)$, t the flow direction and n the direction normal to the flow direction, A the profile area and c the profile chord lenght.

1.3.2 Geometrical constraints

The geometrical constraints are of two types:

- local ones (e.g. two limiting surfaces for the deformation), taken into account by projection:

$$f_1(x_c) \leq \delta x_c \leq f_2(x_c) \tag{1.3}$$

  The final profile can then satisfy the given specifications: e.g. minimal or maximal thickness, unchanged leading or trailing edge, etc..

- global ones (e.g. conservation of a given volume), taken into account by penalty in the cost function, for example:

$$J(x_c) = \alpha\, C_d + \beta\, |V - V^0|/V^0 \tag{1.4}$$

1.3.3 State constraints

The state constraints are taken into account by penalty in the cost function, e.g. a given lift yields:

$$J(x_c) = \alpha\, C_d + \gamma\, |C_l - C_l^0|/C_l^0 \tag{1.5}$$

A special constraint for the case of a fan blade is the inlet-outlet pressure difference. Since the gradient is suitable just for cost functions defined by surface integrals over the shape, we have to express this constraint in function of the aerodynamic coefficients.

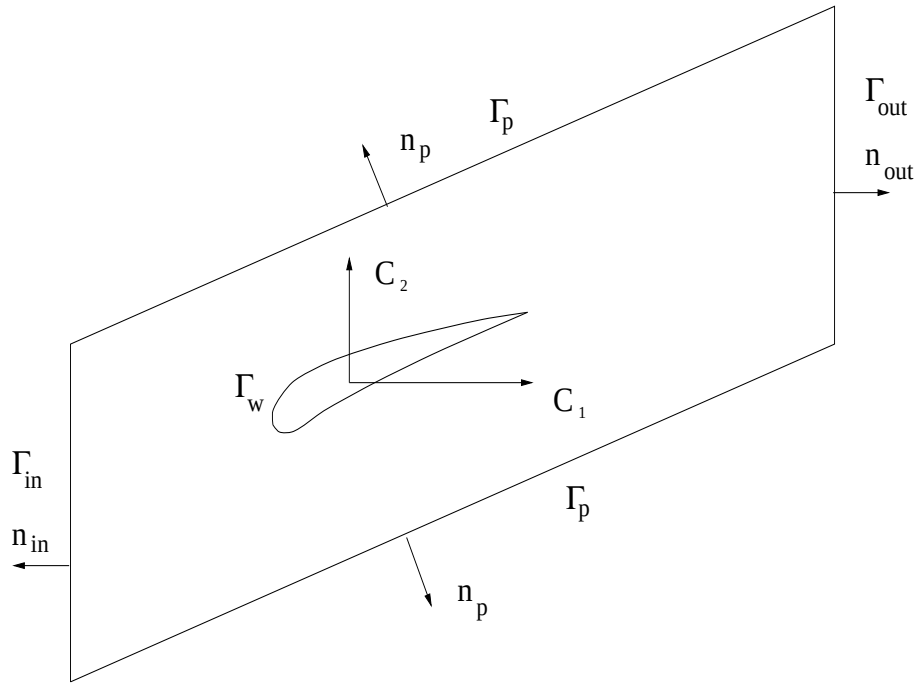

Figure 1.1: The computational domain

For the computational domain Ω having Γ as border shown in Fig. 1.1 we can write the momentum conservation equation for a stationary flow with no external forces:

$$\int_{\Omega} u \nabla u d\Omega = \int_{\Gamma} -pnd\sigma + \int_{\Gamma} (\nu + \nu_t)(\nabla u + \nabla u^T) n d\sigma \tag{1.6}$$

or

$$\int_{\Gamma} u(u.n) d\sigma + \int_{\Gamma} Tnd\sigma = 0 \tag{1.7}$$

We can neglect the viscous effects on the inlet and outlet boundaries. Since on the periodic boundaries all the flow variables are periodic and the normal changes the sign, the sum of the integrals on the periodic boundaries is zero. On the other hand, the advection terms are zero on the profile because the velocity is zero at the wall (no-slip condition). The normals on the inlet boundary and the outlet boundary are the same with the sign changed, therefore we have:

$$\int_{\Gamma_{in}} u(u.n_{in}) d\sigma + \int_{\Gamma_{out}} u(u.n_{out}) d\sigma + \int_{\Gamma_{in}} pn_{in} d\sigma + \int_{\Gamma_{out}} pn_{out} d\sigma + \int_{\Gamma_w} Tn_w d\sigma = 0 \tag{1.8}$$

We consider the average quantities u_{in}, u_{out}, p_{in}, p_{out}:

$$-u_{in}(u_{in}.n_{out})l_{in} + u_{out}(u_{out}.n_{out})l_{out} - p_{in}n_{out}l_{in} + p_{out}n_{out}l_{out} + (C_1 + C_2)\frac{1}{2}\rho_\infty |u_\infty|^2 Ac = 0 \tag{1.9}$$

where C_1 and C_2 are the vectorial aerodynamic coefficients in the horizontal and vertical directions. Since we are interested only in the horizontal components, we can write:

$$-(u_{in}.n_{out})^2 l_{in} + (u_{out}.n_{out})^2 l_{out} + -p_{in}l_{in} + p_{out}l_{out} + C_1\frac{1}{2}\rho_\infty |u_\infty|^2 Ac \tag{1.10}$$

Therefore for $l_{in} = l_{out}$ we obtain:

$$\triangle p = p_{out} - p_{in} = (u_{in}.n_{out})^2 - (u_{out}.n_{out})^2 - C_1\frac{1}{2}\rho_\infty |u_\infty|^2 A\frac{c}{l_{in}} \tag{1.11}$$

The inlet and outlet velocities are constant, therefore the inlet-outlet pressure difference depends only on the horizontal aerodynamic coefficient. In this way the constraints on the inlet-outlet pressure can be introduced in the cost function.

1.3.4 Smoothing operator

The usage of wall nodes coordinates as control parameters, instead of splines, calls for a smoothing of shape deformations. This is done with the smoothing operator:

$$\begin{cases} (I - \zeta\Delta)\delta\tilde{x_c} = \delta x_c \\ \delta\tilde{x_c} = \delta x_c = 0 \qquad \text{when constrained} \end{cases} \tag{1.12}$$

where

- $\delta\tilde{x_c}$ is the smoothed shape variation;
- ζ is the viscosity coefficient in the smoothing process, which is taken proportional to the norm of the second derivatives of the deformation, in order to make the smoothing local (the shape locally smooth remains unchanged).

The smoothing system is solved with Jacobi method iterations (see Ref. [6]).

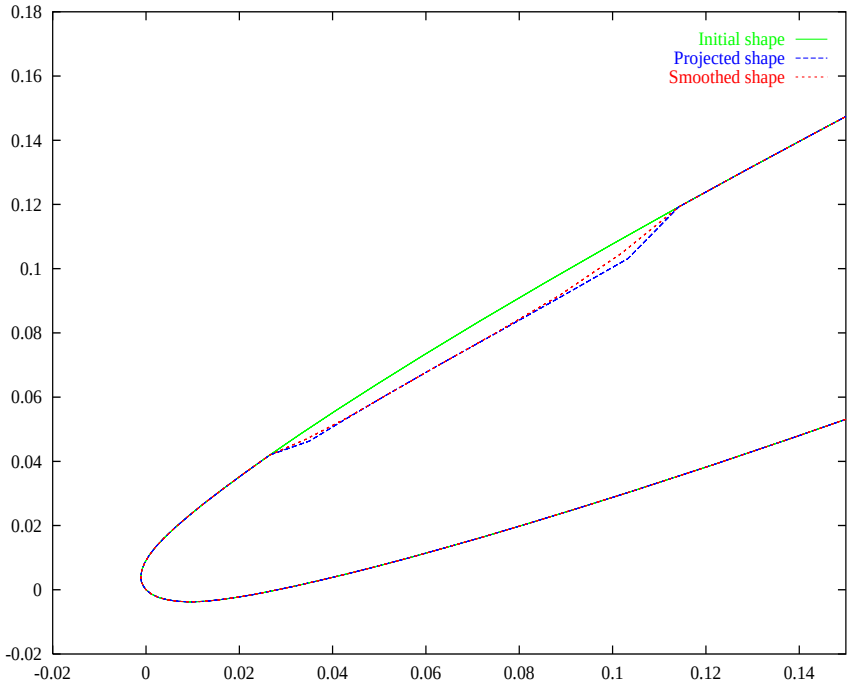

Figure 1.2: Shape smoothing

1.3.5 Mesh deformation

For the new profile generated with the gradient method we have to construct a new mesh. We use an operator with a given deformation on the profile, which doesn't change the mesh triangles orientation, and keeps the same mesh quality (see Ref. [5]).

The deformation of a mesh node is given by:

$$\begin{cases} (\delta x_m)_i = \dfrac{1}{\alpha_i} \displaystyle\sum_{k \in \Gamma_w} w_k \alpha_{ki} \, (\delta \tilde{x}_c)_i \\ \delta \tilde{x}_m = \delta \tilde{x}_c \text{ on } \Gamma_w \end{cases} \tag{1.13}$$

where:

- δx_m is the mesh variation;
- w_k is a weight coefficient for each node k, depending on the neighbouring segments length;
- $\alpha_{ki} = \dfrac{1}{|\tilde{x}_k - \tilde{x}_i|^\beta}$ with β an arbitrary parameter (usually $\beta = 2$);
- $\alpha_i = \displaystyle\sum_{k \in \Gamma_w} w_k \alpha_{ki}$ is a normalisation parameter.

Another mesh deformation technique is to solve a volumic elasticity system of the same form as the smoothing operator:

$$\begin{cases} (I - \eta\Delta)\delta x_m = \delta\bar{x}_m \\ \delta\bar{x}_m = \delta\tilde{x}_c \qquad \text{on } \Gamma_w \\ \delta\bar{x}_m = 0 \qquad \text{in } \Omega \end{cases} \tag{1.14}$$

where:

- δx_m is the mesh variation;
- η is the viscosity coefficient in the mesh deformation process, which is taken proportional to the local element size, in order to avoid mesh degeneration (the mesh practically does not change far away from the profile).

Again, this is made with Jacobi iterations.

The first approach proved to be more robust in treating fine meshes, therefore it was used in all the optimization cases.

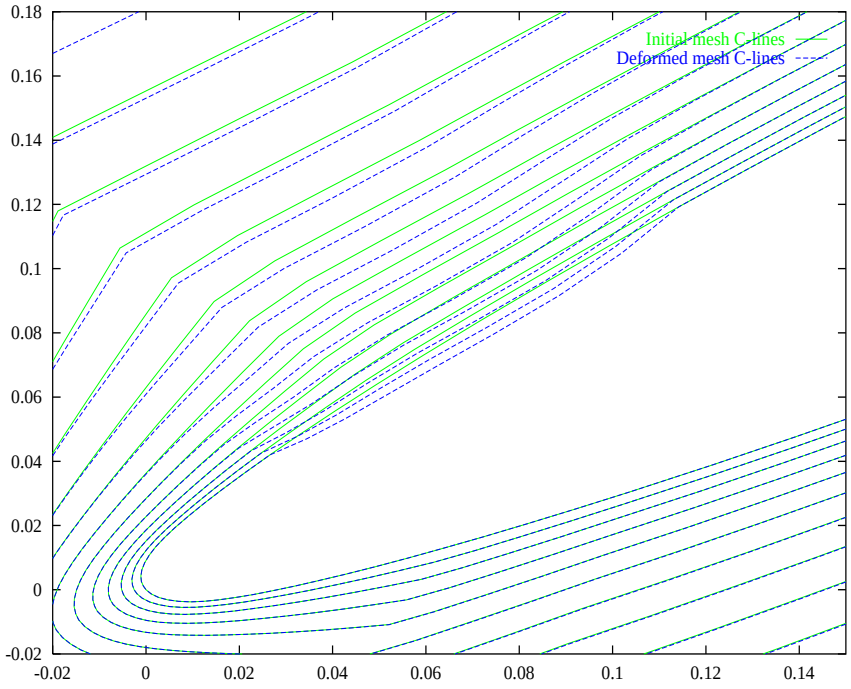

Figure 1.3: Mesh deformation

1.3.6 Optimization algorithm

We can conclude that generally, the optimization code is based on the following algorithm (see Ref. [7, 8, 6]):

Optimization loop

1. Choose initial shape x_c^0 and solve the flow $U^0(x_c^0)$;
2. Compute the gradient $\dfrac{dJ(x_c)}{dx_c}$;
3. Compute the shape deformation δx_c with a gradient method;
4. Smooth the deformations $\delta x_c \to \delta \tilde{x}_c$;
5. Deform the mesh by computing δx_m;
6. compute the new state $U(x_c)$ such that $E(x_c, U(x_c), \nabla_{x_c} U(x_c)) = 0$;
7. if the gradient is converged $\left|\dfrac{dJ(x_c)}{dx_c}\right| < \varepsilon$ stop, else goto step 2.

End of the optimization loop

1.4 Gradient computation

The gradient of the cost function with respect to the control variables is defined as:

$$\frac{dJ}{dx_c} = \frac{\partial J}{\partial x_c} + \frac{\partial J}{\partial x_q}\frac{\partial x_q}{\partial x_c} + \frac{\partial J}{\partial U}\frac{\partial U}{\partial x_q}\frac{\partial x_q}{\partial x_c}. \tag{1.15}$$

where the index q represents all the geometrical quantities (mesh coordinates, shape normals, etc.).

1.4.1 Automatic differentiation

For the computation of the gradient defined as in (1.15) we use automatic differentiation (AD) (see Ref. [9]).

Automatic differentiation is a technique which evaluates the derivatives of some unctions defined by numerical codes with respect to some input variables. The inverse automatic differentiation can be seen as an adjoint method used in optimization problems.

In this approach, the program lines defining the output variables variation and the cost function, including the mesh and the state equations are considered constraints. For each of them Lagrange multipliers p are associated and an augmented Lagrangian L is constructed:

$$L = J + p^t E \tag{1.16}$$

The values p are obtained by imposing that the first order variations of L with respect to the intermediate variables are zero:

$$\frac{\partial L}{\partial U} = \frac{\partial J}{\partial U} + p^t \frac{\partial E}{\partial U} = 0 \tag{1.17}$$

Once the multipliers p are evaluated, the gradient of L can be easily computed:

$$\frac{dJ}{dx_c} = \frac{\partial L}{\partial x_c} = \frac{\partial J}{\partial x_c} + p^t \frac{\partial E}{\partial x_c} \tag{1.18}$$

For this operation we use the automatic differentiator tool Odyssée developed at INRIA. Odyssée differentiates programs written in Fortran 77: for a Fortran procedure describing a function and for a list of independent and dependent variables a Fortran code is generated which evaluates the derivatives of the dependent variables with respect to the independent variables.

1.4.2 Incomplete sensitivities

Using automatic differentiation (Odyssee AD tool) we analyzed the contribution of the different sensitivities to the gradient. In particular, we noticed that when the cost function is based on informations on the shape through boundary integrals (e.g. aerodynamic coefficients), the dominant part in the gradient comes from sensitivities with respect to the geometrical quantities and not to the state. This means that the last term in (1.15) can be dropped:

$$\frac{dJ}{dx_c} \approx \frac{\partial J}{\partial x_c} + \frac{\partial J}{\partial x_q}\frac{\partial x_q}{\partial x_c} \tag{1.19}$$

This avoids the calculation of an adjoint state and decreases significantly the computational cost (see Ref. [6, 8]).

1.5 Optimization methods

1.5.1 Steepest descent method

We use a steepest descent method with fixed step size α for the computation of the new shape (see Ref. [12]):

$$\delta x_c^k = \Pi\left(-\alpha \frac{dJ}{dx_c}(x_c^k)\right), \tag{1.20}$$

where Π is the projection operator on the admissible space, used to take into account local geometrical constraints (two limiting surfaces or curves for the shape variations or original planform kept unchanged). The constant step size is used because for the analyzed 3D problems the line algorithm search becomes too expensive.

1.5.2 Heavy-ball method

The heavy ball method is used to access different minima of the problem, for specific cost functions. This time shape variation is computed as:

$$\begin{cases} (\delta x_c^k)^{sd} = \Pi\left(-\alpha \dfrac{dJ}{dx_c}(x_c^k)\right) \\ \delta x_c^k = \left(\dfrac{\alpha}{\alpha+\varepsilon}\right)(\delta x_c^k)^{sd} + \left(\dfrac{\varepsilon}{\alpha+\varepsilon}\right)\delta x_c^{k-1} \end{cases} \tag{1.21}$$

The coefficient ε is at least an order of magnitude smaller than the step size α. With this method the convergence is faster for oscillating cost functions.

1.5.3 BFGS method

Further on, we investigated the application of an optimization algorithm based on a BFGS method for 3D aerodynamical shape design problems. This method is based on a quadratic approximation of the objective function, which is more effective than a linear approximation (see Ref. [11]).

The search direction for optimization step k:

$$\delta x_c{}^k = \Pi\left(-\alpha^k B^k \frac{dJ}{dx_c}(x_c^k)\right) \tag{1.22}$$

is defined by deflecting the gradient vector with an approximation of the inverse of the Hessian matrix H of the objective function. This approximation B is constructed with the gradient variations evaluated during the optimization process, using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating formula:

$$B^{k+1} = B^k + \left(1 + \frac{\gamma^{k^T} B^k \gamma^k}{\delta^{k^T}\gamma^k}\right)\frac{\delta^k \delta^{k^T}}{\delta^{k^T}\gamma^k} - \left(\frac{\delta^k \gamma^{k^T} B^k + B^k \gamma^k \delta^{k^T}}{\delta^k \gamma^k}\right), \tag{1.23}$$

where $\delta^k = x_c^{k+1} - x_c^k$ and $\gamma^k = \dfrac{dJ}{dx_w}(x_c^{k+1}) - \dfrac{dJ}{dx_c}(x_c^k)$.

Even if the matrix B is constructed to be symmetric and positive definite, an angle condition has to be satisfied between the search direction and the gradient in order to increase the efficiency in the objective function descent. An inexact line search is employed,

with the step size α^k proportional to the gradient modulus, and a continuing update of the hessian is operated.

Although the global convergence of such a scheme requires stringent conditions and therefore the decreasing of the objective function is not a priori guaranteed, the local convergence rate is often asymptotically superlinear.

1.6 Turbomachinery Flows

1.6.1 Blade cascade

We consider a complete configuration of an engine cooling axial fan. For simplicity reasons, we reduce the 3D case at a simpler 2D case.

The most general fan data are the geometry and the inlet flow characteristics for a given operating point. We suppose that the fan is defined by the blade number N, by the tip radius R_{min} an the hub radius R_{max}, and by the blade geometry (profile, twist angle, chord, leading edge, trailing edge). The flow is defined by the flow rate Q and by the rotational velocity Ω.

Because the fan axial symmetry and the rotational flow characteristics, the flow can be assumed to be the same for all the blades and to be periodic in the circumferential direction. Considering a three-dimensional cylindrical computational domain for the fan, we choose a characteristic section for the two-dimensional computation at a radius R. We obtain a 2D blade cascade (fig. 1.4), which is reduced at a single domain around the profile because of the angular periodicity.

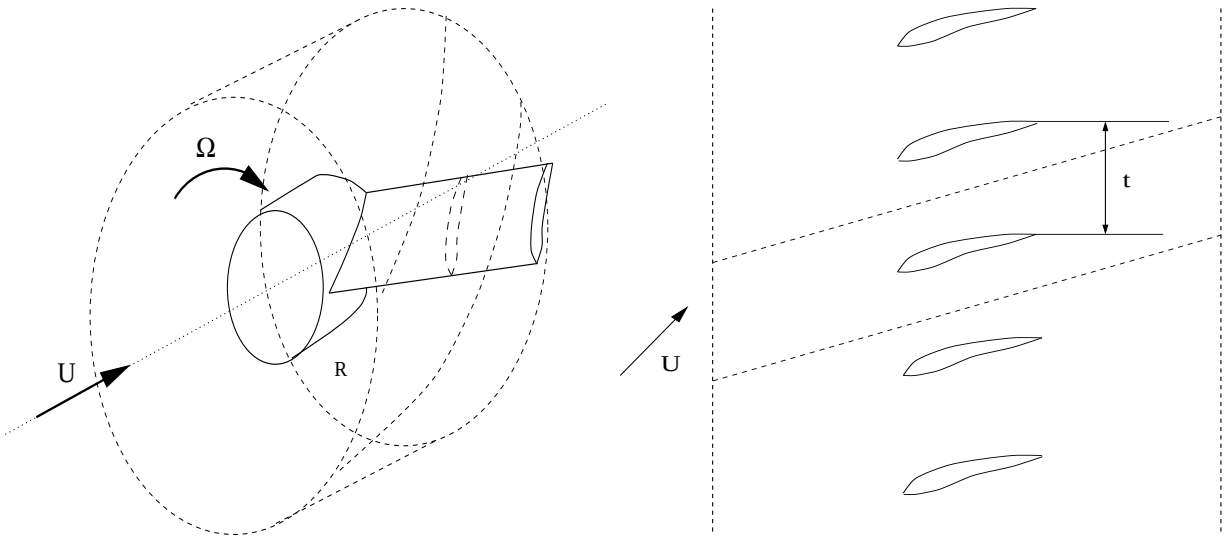

Figure 1.4: Blade cascade

1.6.2 2D case

The pitch in the angular direction of the computational domain is imposed by the radius R and the blade number N: $t = \dfrac{2\pi R}{N}$. The axial dimension must be big enough to satisfy at best the uniform flow conditions at the inlet and at the outlet (difficult due to the wake).

The geometry of the computational domain (fig. 1.5) is defined by the parameters (see Ref. [4, 2]):

- the chord lenght c and the tangential spacing between blades (pitch) t;
- the solidity $\sigma = \dfrac{c}{t}$;
- the twist angle λ between the chord and the axis;
- the angles β_1, β_2, β_∞ between the flow directions and the axis;
- the angle of incidence $\alpha = \beta_1 - \lambda$;
- the deflexion angle $\theta = \beta_2 - \beta_1$.

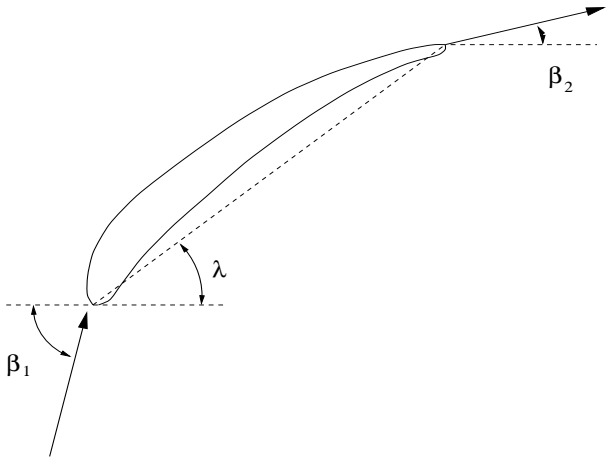

Figure 1.5: Profile geometry

The inlet conditions are given by the flow rate and the fan rotational velocity:

$$\begin{cases} U_x = \dfrac{Q}{\pi(R_{max}^2 - R_{min}^2)} \\ U_t = \Omega R \end{cases} \tag{1.24}$$

The aerodynamic coefficients are defined on the flow direction at the infinity.

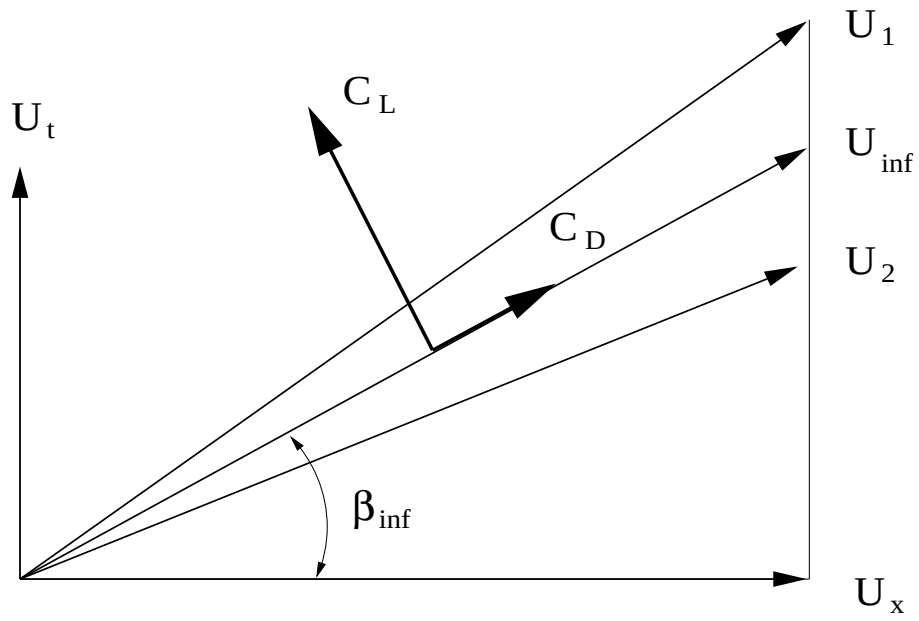

Figure 1.6: Velocity vectors

1.6.3 Fan efficiency

The fan efficiency is defined as:

$$\eta = \frac{Q \triangle p}{\Omega M} \tag{1.25}$$

where:

- $\triangle p$ is the inlet-outlet pressure difference;
- M is the torque resistance momentum.

1.6.4 Fan optimization criteria

For the real case of a 3D fan configuration we are interested in increasing the efficiency $\eta = \frac{Q \triangle p}{\Omega M}$, which means to increase the inlet-outlet pressure difference $\triangle p$ and decrease the torque resistance momentum M.

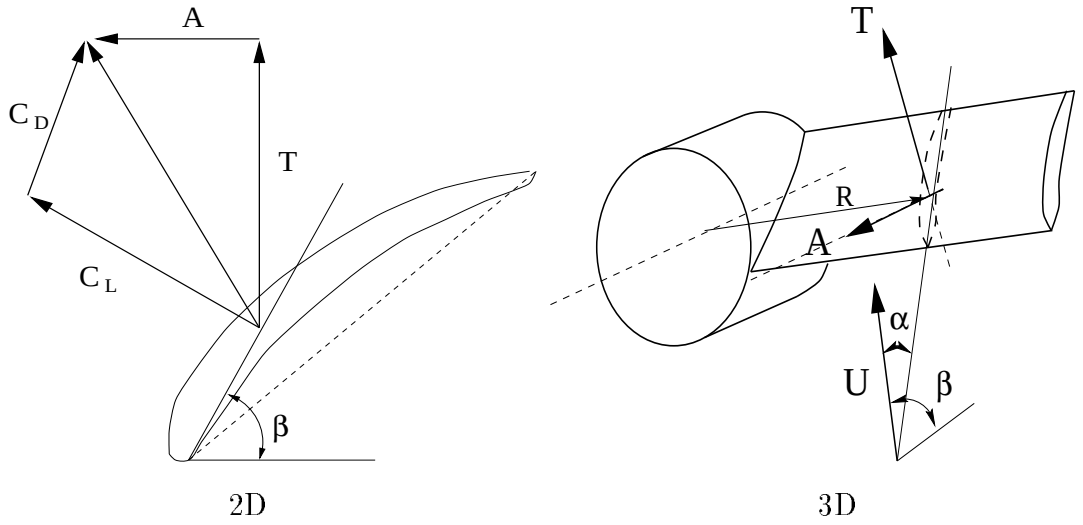

Figure 1.7: Aerodynamic forces on the blade

The inlet-outlet pressure difference depends on the horizontal aerodynamic coefficient (see 3.4.2) and the torque resistance momentum depends on the tangential force (fig. 1.7):

$$\begin{cases} A = C_l\, sin\beta - C_d\, cos\beta \\ T = C_l\, cos\beta + C_d\, sin\beta \end{cases} \tag{1.26}$$

Therefore the cost function should be $j(x_c) = \dfrac{T}{A}$. A simple analysis of this function shows that its minimum corresponds to a C_d/C_l minimum. Consequently, we consider the following cost function:

$$j(x_c) = \frac{C_d}{C_l} \tag{1.27}$$

Chapter 2

TASCflow Environment

2.1 General description

In the most general terms, the standard phases of solving a CFD problem in the TASCflow environment are the following:

1. **Grid Generation**:

   First, the creation of the computational grid over which the analysis will be performed. This phase is performed using programs under Applications, such as TASCgrid or ICEM, or the user can read in meshes from many other sources, such as CAD packages.

2. **Pre-Processing**:

   Next is the pre-processing stage performed by TASCbob that includes specifying the boundary conditions of the problem, the properties of the material which is modelled, and the physical zones of the model (e.g. what parts are stationary and what parts are in rotating reference frames).

3. **Solving**:

   Then, the solver TASCflow3D is used to compute the flow within the computational domain. This includes setting up the solver control parameters that are used to drive the convergence to a stationary solution.

4. **Post-Processing**:

   After getting a flow solution, the results are examined with TASCtool. Both flow visualization, i.e. graphical representation of the fluid flow, and quantitative calculation such as efficiency over the whole domain can be performed.

5. **Modification**:

Lastly, and usually after post-processing the results, some aspect of the problem geometry, grid, boundary conditions, solver parameters, etc., will be adjusted and the CFD run is repeated until the desired accuracy is achieved.

From the point of view of using the optimization code in the TASCflow environment for mesh generation and flow resolution, we can see that the grid generation (using TASCgrid and different templates) is used only in the initial phase of the optimization loop (unless we do not perform mesh regeneration at each optimization iteration instead of mesh deformation). Afterwards, as we have seen previously in the optimization algorithm (1.3.6), the optimization code TASCOPT-3.0 scheme uses only the code optimtasc.exe and the fluide solver TASCflow3D.

Therefore, optimtasc.exe needs only the solution and the grid (including topology, boundary conditions, etc.), which are read, processed and saved in the same format.

2.2 TASCflow capabilities

TASCflow is an integrated and software system capable of solving diverse and complex multi-dimensional fluid flow problems. The fluid flow solver, TASCflow3D, provides solutions for incompressible or compressible, steady-state or transient, laminar or turbulent single-phase fluid flow in complex geometries. The software uses block-structured non-orthogonal grids with grid embedding and grid attaching to discretize the domain.

The software system has additional capabilities which can predict subsonic, transonic and supersonic compressible flows including temperature solutions in solid regions of the domain, for laminar or turbulent flow. As well, CFX-TASCflow has the additional capability of the Eddy Dissipation Model for the simulation of turbulent reacting fluid mixtures of separate physical components or species, and the Lagrangian Tracking Model for calculation of discrete particle trajectories (passive or with reaction) in a hydrodynamic flow field.

2.2.1 Governing equations

Instantaneous equations

For a single species Newtonian fluid, in a Cartesian coordinate system, the conservation equations for mass, momentum and energy may be expressed in tensor form (with the Einstein summation convention), as:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial}{\partial x_j}(\rho u_j) = 0 \\ \dfrac{\partial}{\partial t}(\rho u_i) + \dfrac{\partial}{\partial x_j}(\rho u_j u_i) = -\dfrac{\partial P}{\partial x_i} + \dfrac{\partial \tau_{ij}}{\partial x_j} + S_{ui} \\ \dfrac{\partial}{\partial t}(\rho H) - \dfrac{\partial P}{\partial t} + \dfrac{\partial}{\partial x_j}(\rho u_j H) = -\dfrac{\partial q_j}{\partial x_j} + \dfrac{\partial}{\partial x_j}(u_i \tau_{ij}) + S_E \end{cases} \tag{2.1}$$

In the above equations u_i represents the velocities in the x_i-coordinate directions, P is the static pressure, H is the total enthalpy, ρ is the density, τ_{ij} is the viscous stress tensor, q_j is the molecular energy transport due to conduction, and the S terms are additional source terms.

The total enthalpy is defined as: $H = h + \dfrac{u_i u_j}{2}$, where h is the static enthalpy of the fluid.

The molecular fluxes τ_{ij} and q_j for a newtonian fluid are expressed in terms of velocity, temperature and concentration gradients (Stokes's law and Fourier and Fick's law):

$$\begin{cases} \tau_{ij} = \mu \left(\dfrac{\partial u_i}{\partial x_j} + \dfrac{\partial u_j}{\partial x_i} \right) + \dfrac{2}{3}\mu \dfrac{\partial u_l}{\partial x_l}\delta_{ij} \\ q_j = -\lambda \dfrac{\partial T}{\partial x_j} - \sum_k^n \Gamma_k h_k \dfrac{\partial Y_k}{\partial x_j} \end{cases} \tag{2.2}$$

where μ is the dynamic viscosity of the fluid, λ its conductivity and Γ_k, h_k and Y_k are the molecular diffusion coefficient, static enthalpy and mass fraction of species k, respectively. The second term on the right hand side of equation defining q represents energy diffusion due to molecular diffusion when the fluid components have different enthalpies.

Turbulence model

In turbulent flows, the value of scalar variables fluctuates and the instantaneous value of any scalar may be expressed as the sum of a mean and a fluctuating component. The fluctuating components are not solved directly in CFX-TASCflow, and therefore, it is necessary to express the fluctuating values in terms of the corresponding mean values.

The instantaneous scalar values in Navier-Stokes equations are expressed in terms of mean and fluctuating components through a process of time-averaging, called Reynolds-Stress averaging in the momentum equations. Each dependent variable in the original conservation equations is decomposed into a mean $\overline{\phi}$ and fluctuating component ϕ': $\phi = \overline{\phi} + \phi'$, the mean component of $\overline{\phi}$ being $\overline{\phi} = \dfrac{1}{\triangle t} \int_t^{t+\triangle t} \phi dt$, where the time interval $\triangle t$ is long compared to the time scale of the turbulent fluctuations. For compressible flows, it

is often useful to perform a mass weighted decomposition called Favre averaging: $\tilde{\phi} = \frac{\overline{\rho\phi}}{\overline{\rho}}$; the fluctuating component is given a double prime: $\phi = \tilde{\phi} + \phi''$.

The Reynolds stress and turbulent flux terms are related to the mean flow variables using an eddy viscosity assumption:

$$\overline{\rho u_i'' u_j''} = -\mu_t \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right) + \frac{2}{3} \left(\mu_t \frac{\partial \tilde{u}_l}{\partial x_l} + \overline{\rho} \tilde{k} \right) \delta_{ij} \tag{2.3}$$

This equation reduces to the time averaged variant used when a fluid is modeled as incompressible:

$$\overline{\rho u_i'' u_j''} = \overline{\rho u_i' u_j'} = -\mu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) + \frac{2}{3} \rho k \delta_{ij} \tag{2.4}$$

The fluctuating viscous work term of Eq. (2.2) for homogeneous isotropic turbulence is approximated by:

$$\frac{\partial (\overline{u_i'' \tau_{ij}''})}{\partial x_j} \approx \overline{\frac{\partial}{\partial x_j} \left(\mu_t \frac{\partial \tilde{k}}{\partial x_j} \right)} \tag{2.5}$$

Equations (2.4) and (2.5) can only express the turbulent fluctuation terms as functions of the mean variables if the turbulent kinetic energy, k, and turbulent viscosity, μ_t, are known. In CFX-TASCflow the $k - \varepsilon$ turbulence model (Launder and Spalding, see Ref. [3]) provides these variables.

The $k - \varepsilon$ model was first derived for incompressible flows in which the density fluctuations can be ignored. Current practice has also been to use the resulting model for compressible flows. Consequently, the assumption that the mass weighted and time averaged variants of the turbulent kinetic energy are equivalent is made, i.e. $\tilde{k} = \overline{k} = k$.

According to Boussinesq's assumption, the eddy viscosity μ_t is calculated from the product of a turbulence velocity scale ($\alpha\sqrt{k}$) and turbulence length scale ($\alpha k^{3/2}/\varepsilon$) as:

$$\mu_t = \rho c_\mu \frac{k^2}{\varepsilon} \tag{2.6}$$

where ε is the dissipation rate of k and c_μ is a model constant.

Local values of k and ε are obtained from the solution of the following semi-empirical transport equations (the equation for k is derived from the transport of Reynolds stresses, and the equation for ε is assumed to be similar to the equation for k empirically):

$$\begin{cases} \dfrac{\partial(\rho k)}{\partial t} + \dfrac{\partial(\rho \overline{u}_j k)}{\partial x_j} = \dfrac{\partial}{\partial x_j}\left(\Gamma_k \dfrac{\partial k}{\partial x_j}\right) + P_k - \rho\varepsilon \\ \dfrac{\partial(\rho \varepsilon)}{\partial t} + \dfrac{\partial(\rho \overline{u}_j \varepsilon)}{\partial x_j} = \dfrac{\partial}{\partial x_j}\left(\Gamma_\varepsilon \dfrac{\partial \varepsilon}{\partial x_j}\right) + \dfrac{\varepsilon}{k}(c_{\varepsilon 1} P_k - \rho c_{\varepsilon 2}\varepsilon) \end{cases} \tag{2.7}$$

where the diffusion coefficients are given by:

$$\begin{cases} \Gamma_k = \mu + \dfrac{\mu_t}{\sigma_k} \\ \Gamma_\varepsilon = \mu + \dfrac{\mu_t}{\sigma_\varepsilon} \end{cases} \tag{2.8}$$

The production rate of turbulent kinetic energy P_k is given by:

$$P_k = -\overline{\rho u_i' u_j'} \frac{\partial \overline{u}_i}{\partial x_j} = -\mu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) \frac{\partial \overline{u}_i}{\partial x_j} - \frac{2}{3}\left(\rho k + \mu_t \frac{\partial \overline{u}_l}{\partial x_l}\right) \frac{\partial \overline{u}_k}{\partial x_k} \tag{2.9}$$

The standard $k-\varepsilon$ model for the production term, mathematically valid for incompressible flow, can be written as:

$$P_k = -\mu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) \frac{\partial \overline{u}_i}{\partial x_j} \tag{2.10}$$

Eq. (2.9) reduces to Eq. (2.10) for incompressible flows, and for compressible flows the difference will only be large in regions with high velocity divergences, such as in shocks. The default model for the production term is that of Eq. (2.10).

The values of all the constants in the model are calibrated for the flat plate test case:

$$c_\mu = 0.09 \qquad c_{\varepsilon 1} = 1.44 \qquad c_{\varepsilon 2} = 1.92 \qquad \sigma_k = 1.0 \qquad \sigma_\varepsilon = 1.3 \qquad Pr_t = 0.9 \tag{2.11}$$

Mean form of equations

With the preceeding closure of the turbulence model, the final form of the mean flow equations can be stated. At this stage, since all the variables are mean flow quantities, it is customary to drop the superscripts, time and Favre averaging symbols.

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial}{\partial x_j}(\rho u_j) = 0 \\ \dfrac{\partial}{\partial t}(\rho u_i) + \dfrac{\partial}{\partial x_j}(\rho u_j u_i) = -\dfrac{\partial P}{\partial x_i} + \dfrac{\partial}{\partial x_j}\left\{\mu_{eff}\left(\dfrac{\partial u_i}{\partial x_j} + \dfrac{\partial u_j}{\partial x_i}\right)\right\} + S_{ui} \\ \dfrac{\partial}{\partial t}(\rho H) - \dfrac{\partial P}{\partial t} + \dfrac{\partial}{\partial x_j}(\rho u_j H) = -\dfrac{\partial}{\partial x_j}\left(\lambda \dfrac{\partial T}{\partial x_j} + \dfrac{\mu_t}{Pr_t}\dfrac{\partial h}{\partial x_j}\right) + \\ + S_E \dfrac{\partial}{\partial x_j}\left\{u_i\left[\mu_{eff}\left(\dfrac{\partial u_i}{\partial x_j} + \dfrac{\partial u_j}{\partial x_i}\right) - \dfrac{2}{3}\mu_{eff}\dfrac{\partial u_l}{\partial x_l}\delta_{ij}\right] + \mu \dfrac{\partial k}{\partial x_j}\right\} \end{cases} \tag{2.12}$$

where $\mu_{eff} = \mu + \mu_t$, $H = h + \dfrac{1}{2}u_i u_i + k$.

For compressible flows, density is a time-average and velocity is a mass-average. For incompressible flows, density is constant, the transient term vanishes and the mean velocity is a time-average.

For flows in rotating frames of reference, the effects of the Coriolis and Centrifugal forces are modeled in the code. In this case:

$$\vec{S}_u = -2\vec{\Omega} \times \vec{U} - \vec{\Omega} \times (\vec{\Omega} \times \vec{r}) \tag{2.13}$$

where $\vec{\Omega}$ is the rotational speed and $\vec{r}$ is the location vector.

In a rotating frame of reference, the rothalpy, I, is convected in place of the total enthalpy, H, in the energy equation:

$$I = H - \frac{\omega^2 R^2}{2} \tag{2.14}$$

where ω is the rotational speed and R is the local radius. Since the rotational energy is not included in the transient term, rothalpy is only conserved in the steady solution if the rotational speed is constant.

2.2.2 Discretization

CFX-TASCflow is a Finite Volume method, but is based on a Finite Element approach of representing the geometry. Thus the method used in CFX-TASCflow retains much of the geometric flexibility of Finite Element methods as well as the important conservation properties of the Finite Volume method.

The Finite Volume method proceeds by integrating the governing equations over a fixed control volume using Gauss's Theorem. The computational domain is discretized into elements, then control volume surfaces are defined by element mid-planes.

This approach has been used by other researchers in the field. The procedure creates a control volume for each node, with the boundary of each interior control volume defined by eight line-segments in 2D and 24 quadrilateral surfaces in 3D.

The integral equations are applied to each discrete control volume created by this technique. The continuous volume integrations are relatively easy to convert to a discrete form. The continuous surface integrations are more involved and are converted to a discrete form by evaluating them at what are called "Integration Points" (*ip* for short). In 3-D, the flux element consists of 8 octants and 12 integration point surfaces, each containing an integration point location. The surface fluxes must be discretely represented at the integrations points to complete the conversion of the continuous equations to their discrete counterparts.

The discrete form of the integral equations are written as:

$$\begin{cases} \rho V \left(\dfrac{\rho - \rho^0}{\triangle t} \right) + \sum\limits_{ip} (\rho u_j \triangle n_j)_{ip} = 0 \\ \rho V \left(\dfrac{u_i - u_i^0}{\triangle t} \right) + \sum\limits_{ip} \dot{m}_{ip} (u_i)_{ip} = \\ \sum\limits_{ip} (P \triangle n_i)_{ip} + \sum\limits_{ip} \left(\mu_{eff} \left(\dfrac{\partial u_i}{\partial x_j} + \dfrac{\partial u_j}{\partial x_i} \right) \triangle n_j \right)_{ip} + \overline{S}_{u_i} V \\ \rho V \left(\dfrac{\phi_i - \phi_i^0}{\triangle t} \right) + \sum\limits_{ip} \dot{m}_{ip} \phi_{ip} = \sum\limits_{ip} \left(\Gamma_{eff} \dfrac{\partial \phi}{\partial x_j} \triangle n_j \right)_{ip} + \overline{S}_{\phi} V \end{cases} \tag{2.15}$$

where $\dot{m}_{ip} = (\rho u_j \triangle n_j)^0_{ip}$ and V is the volume of the control volume, the subscript ip denotes an integration point, the summation is over all the integration points of the surface, $\triangle n_j$ is the discrete outward surface vector, $\triangle t$ is the time step, the superscripts 0 mean "at the old time level", and the overbar on the source terms indicate an average value for the control volume.

More details can be found in TASCflow Theory - User Documentation (see Ref. [1]).

2.2.3 Boundary conditions

TASCflow allows a large variety of boundary conditions. We only present the ones used in our specific computations (see Ref. [1]).

No-slip walls

A no-slip wall exists where a viscous flow is in contact with a solid object. Right at the wall the fluid is stationary with respect to the solid object. Only impermeable walls are considered.

The fluid velocity normal to the wall is zero:

$$u_i n_i|_b = 0 \tag{2.16}$$

The boundary wall momentum flow includes pressure and viscous forces:

$$F_i^{wall}|_{ip} = F_i^{press} + F_i^{visc} \tag{2.17}$$

The pressure force is computed as the product of the boundary area A_i and the pressure p_i: $F_i^{press} = A_i p_{ip}$ and the normal viscous force is set to zero such that: $F_i^{visc} = F_i^{tang}$. Two different evaluations of F_i^{tang} are available.

For flows where near the wall the velocity gradients are solved (laminar flow, and the 2-layer turbulence model), F_i^{tang} is estimated in terms of the known wall velocity and the control volume velocities in the element containing the boundary face: $F_i^{visc}|_{ip} = A_j \tau_{ij}|_{ip}$, where $\tau_{ij}|_{ip}$ is computed at each ip location from $\tau_{ij}|_{ip} = \mu_{ip} \left(\dfrac{\partial u_i}{\partial x_j} + \dfrac{\partial u_j}{\partial x_i} \right)_{ip}$

For flows where the grid near the wall is too coarse to adequately resolve the velocity gradients based on linear variation within the element, a logarithmic velocity profile is assumed to apply between the wall and the edge of each boundary control volume on the boundary face. Based on this assumed profile, the wall shear stress is computed and hence the wall tangential viscous force. The boundary shear stress is computed at each boundary ip location, consistent with the assumed logarithmic velocity profile, the known wall velocity, and the control volume velocities in the element containing the boundary face. The log-law velocity profile model requires as input the fluid velocity near the wall, u_t, and the distance away from the wall, $\triangle n$, at which u_t is estimated (see Ref. [1]).

An alternative to the use of wall functions is to employ a turbulence model that can more closely resolve the near-wall region. One such model is known as the two-layer model. Rodi (see Ref. [13]), provides a summary of experience with two-layer models that combine the $k - \varepsilon$ model with a one-equation model in the near-wall region. Patel et al. (see Ref. [10]), provide a review of several near-wall and low-Reynolds number turbulence models. Two layer models divide the computational domain into two regions: away from walls and near walls. The standard $k - \varepsilon$ model is used away from the wall. In the near wall region, a one-equation model is employed to establish the turbulent kinetic energy while the length scale is specified by reasonably well-established algebraic equations.

When the $k - \varepsilon$ turbulence model is used in conjunction with a log-law velocity profile or when using the 2-layer model, the flux of k and ε through the wall is assumed to be zero. The production of turbulent kinetic energy is estimated in the near wall region, based on an assumption of local equilibrium between the production and dissipation of turbulent kinetic energy.

Inlets

An inlet boundary condition is one where the fluid enters the domain. There are many different types of inlet boundary condition combinations for the mass and momentum equations.

The boundary velocity can be specified with a non-zero component into the domain $(u_i)_{spec}$ and the total boundary mass flow is $\dot{m}_{total}$ is specified. The mass flow per unit boundary surface area, $(\rho u_n)_{spec}$, is computed from $\dot{m}_{total}$ and the boundary surface: $(\rho u_n)_{spec} = \dfrac{\dot{m}_{total}}{\int_S dA}$, where the integral of the area is over the entire boundary surface S through which $\dot{m}_{total}$ is specified. The value $(\rho u_n)_{spec}$ is held constant over the entire boundary surface. The flow direction D_i is also specified, which constrains the inlet velocity u_i such that: $u_i D_i|_b = (u_i u_i)^{1/2}|_b$.

The pressure can be specified in a number of different ways at an inlet (static pressure, total pressure, average pressure, etc.): $p_b = p_{spec}$, or $p_{total}|_b = p + \dfrac{1}{2}\rho u_i u_i|_b$.

In all of these cases, the boundary mass flow is an implicit result of the flow simulation.

The inlet static temperature is specified: $T_b = T_{spec}$, or the inlet total temperature is specified: $T_{total} = T + \dfrac{u_i u_i}{2c_p}$.

The inlet turbulence quantities are usually expressed in terms of the turbulence intensity, T_u, and the energy-containing eddy length scale, L_ε, where: $k_b = \dfrac{3}{2}T_u^2 u_i u_i|_b$ and $\varepsilon_b = \dfrac{k_b^{3/2}}{L_\varepsilon}$.

For our flow simulation we imposed for the inlet the velocity u, the flow direction D, the turbulent energy k and the dissipation rate ε.

Outlets

An outlet boundary condition is one where the fluid exits the domain. The hydrodynamic boundary condition specification usually involves some constraint on the boundary static pressure.

The total mass flow m_{total} can be specified. The mass distribution across the outlet is determined by weighting according to the local mass distribution.

The boundary velocity can be specified with a non-zero component out of the domain: $(u_i)_{spec}$.

Consistent with a fully developed flow at an outlet, the normal viscous force component is assumed to be zero. The outlet pressure is evaluated by one of the following constraints.

The outlet pressure is specified: $p_b = p_{spec}$, over the whole outlet boundary or only at a given location, or the outlet pressure can be constrained such that the average pressure is a specified value: $\frac{1}{A}\int_A pdA = p_{average}$, where the integral is over the entire outlet boundary surface, and A is the total boundary surface area.

For our flow simulations we imposed for the outlet the pressure p.

Openings

An opening boundary condition allows the fluid flow to cross the boundary surface in either direction. For example all of the fluid might flow into the domain at the opening, or all of the fluid might flow out of the domain, or a mixture of the two might occur, with some fluid flowing into the domain over part of the opening and some fluid flowing out of the domain over the remainder of the opening. An opening condition might be used where it is known that the fluid flows in both directions across the boundary.

The static or total pressure and direction can be specified at an inlet, and the static pressure can be specified at an outlet. Two different combinations are available as an opening condition:

1. Specify the static pressure and direction where the fluid enters the domain, and specify the static pressure (only) where the flow exits the domain. It is also possible to leave the direction unspecified at the inlet (in which case the outlet static pressure condition is applied unchanged to the case of fluid flowing into the domain). This condition is most stable when the fluid is flowing approximately parallel to the boundary (the normal component being less than the tangential component), such as at a far field boundary.

2. Specify the total pressure and direction where the fluid enters the domain, and specify the static pressure where the flow exits the domain. This condition is most stable when the fluid is flowing approximately normal to the boundary (the normal component is greater than the tangential component), such as where an outlet region is located in the middle of a recirculation zone.

Connections between boundary surfaces

Connection conditions refer to the following boundary conditions:

1. Periodic boundary conditions, where one boundary surface is connected to another boundary surface. Periodic boundary conditions are used to define the domain around a blade for the case of linear cascade of blades in turbomachinery.

2. Grid embedding/attaching interfaces:

(a) embedding interfaces: where one grid is connected around the perimeter of a host grid as a result of a grid embedding operation.

(b) attaching interfaces: where two grids are connected over a surface as a result of a grid attaching operation.

The surfaces involved in the connection between the two grids touch one another. In general, the connection is not one-to-one (e.g. many faces on an embedded grid connect to fewer faces on the host grid).

2.3 Different meshes used in TASCOPT-3.0

The Advanced Scientific Computing Ltd. TASCgrid software generates 2–D and 3–D multi-blocks structured computational grids that can be used in the numerical solution of fluid flow and heat transfer problems. The grids generated are boundary-fitted and (in general) non-orthogonal and curvilinear.

The overall process of grid generation involves important preparatory work as well as direct use of the computer (i.e. running TASCgrid). The grid generation process involves the following four steps:

1. Translate the base geometric data into a precise mathematical description.
2. Decide on the shape and number of nodes of the grid.
3. Generate a computer representation of the geometry (Geometry Phase: define geometry of the physical domain).
4. Assign nodes to the domain (Curve, Surface and Interior Phase: define the layout of nodes at corners, distribute nodes on surfaces in 3–D, distribute nodes in the interiors of regions).

Inputs to the programs are in the form of text files. Information is passed between the programs by means of data files. The product of the final program (TASCgridi) is a file containing the coordinates of the finished grid.

The computational aspects of grid generation have been divided into four phases. TASCgrid is implemented as four programs corresponding to these phases. To generate a grid, the user must run each of the programs in sequence. Only three of these programs are required to generate a 2–D grid.

TASCgridg provides a means for describing the physical domain. Ideally, the geometric description should be independent of how the grid is to be attached, helping to simplify the task of grid refinement or adjustment. TASCgridg can be considered as similar to a simple Computer Aided Design (CAD) surface modelling package.

The layout of the grid is defined in TASCgridc, where the first grid nodes are attached to corners of the physical domain. It is in the input to TASCgridc that the numbers of grid nodes (ID, JD and KD) are defined. With care, the input files can be constructed such that the grid can be refined or coarsened by changing only these three numbers. Also in TASCgridc, nodes are distributed along the edges of the domain.

TASCgrids and TASCgridi are designed to construct the grid in a number of separate 'regions'. The user must decide how best to divide the physical domain into regions. Each region is normally assigned to a computational rectangle in 2–D or to a cuboid in 3–D, although the code also has a limited ability to handle regions of a more general shape.

The grid in each region may be generated either with one of three forms of algebraic interpolation (transfinite, semi-isogeometric, or isogeometric), or with an elliptic interpolation scheme. The former is simpler and therefore quicker in execution, while the latter is more flexible but slower. Algebraic interpolation obtains the location of each interior node from the locations of the exterior boundary nodes using an algebraic formula. Elliptic grid generation involves solving a coupled set of transformed Poisson partial differential equations.

Each program requires input from the previous program(s) and from the terminal. The program will then create graphical and textual output for user to analyze and verify, as well as output needed for running the next program(s).

The optimization scheme has been validated in several steps going from the simplest topology with one single block to the most complicated general case, as we will see in the next sections.

2.3.1 Mono-block grids

We present gradually the grids used in our computations. There are basically two kinds of grids around the profile used by us in the TASCflow environment. This region around the profile is the most important for a good resolution of the flow and care has to be taken in order to generate a good grid for each specific case. For multiblock grids, simple blocks were added around this core region.

The C-grid has the topological lines disposed in C around the profile and is best suited for profiles with a sharp trailing edge. The O-grid has the topological lines disposed in O and therefore is suited for profiles with a round trailing edge.

We show how the grids are constructed, presenting the advantages and the defaults of each type.

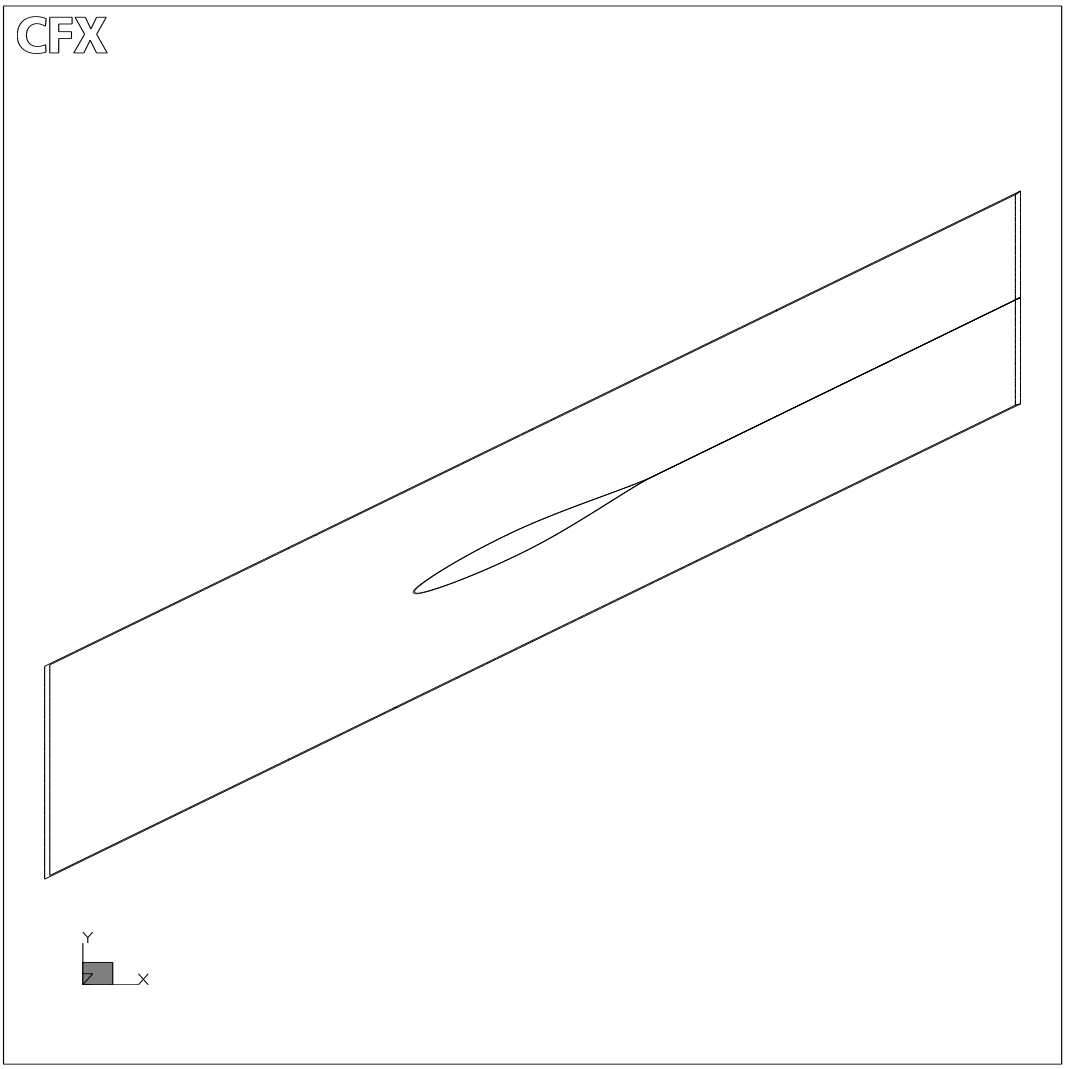

Figure 2.1: Topological lines for a C-grid

In TASCflow, the 2–D computations are in fact 3–D computations performed on grids with the Z-index $KD = 3$, having the same 2–D topology and node distribution for every K.

This grid is realized by the simplest template and has the advantage that is quickly generated and is easy to use. However, this type of grid is quite constraining and becomes easily unuseful for highly cambered profiles and high stagger angles.

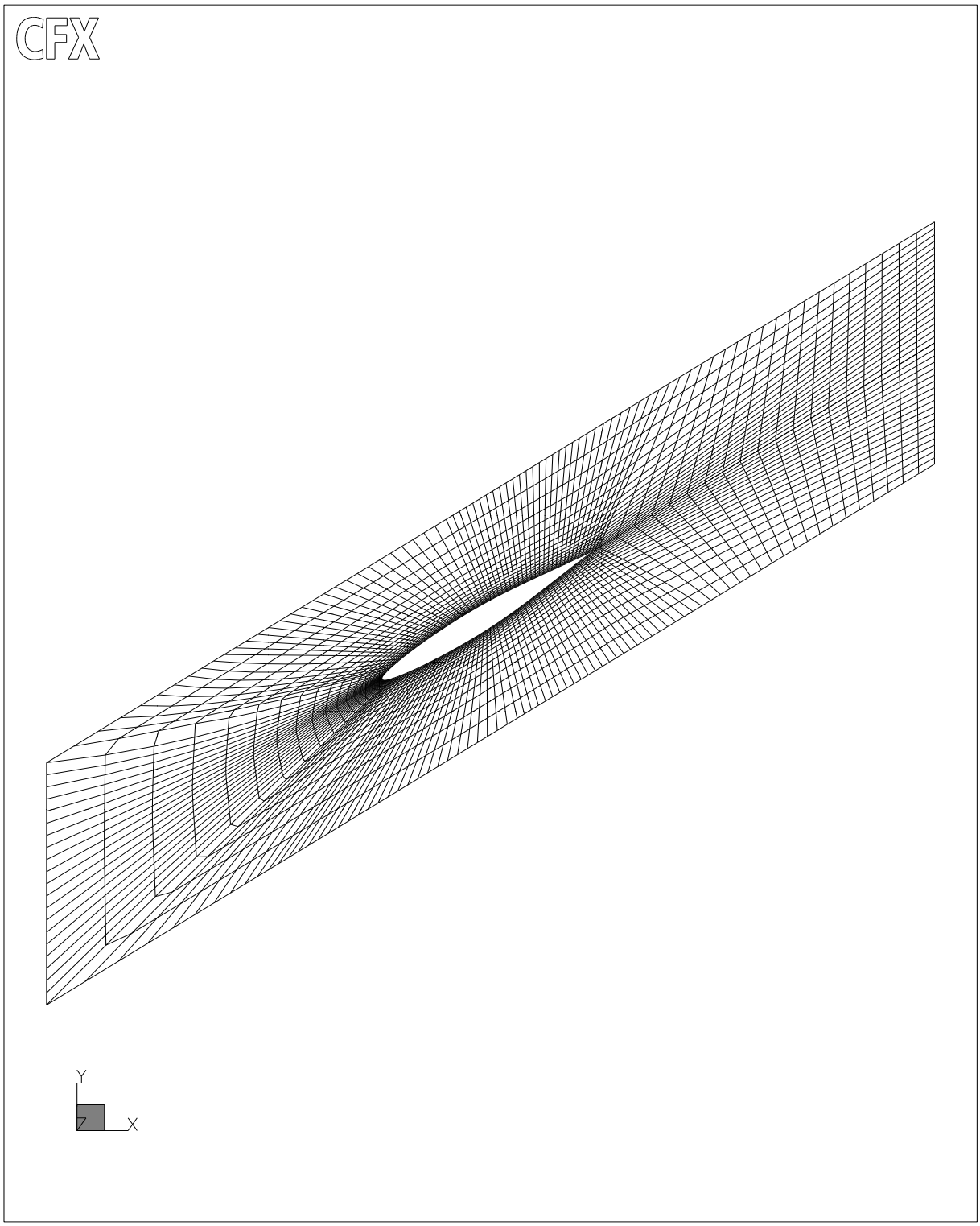

Figure 2.2: C-grid

The grid I-index lines are disposed in C around the profile, allowing a good discretization of the trailing edge. The boundary layer is modelized by a geometrical ratio nodes distribution on the J-index.

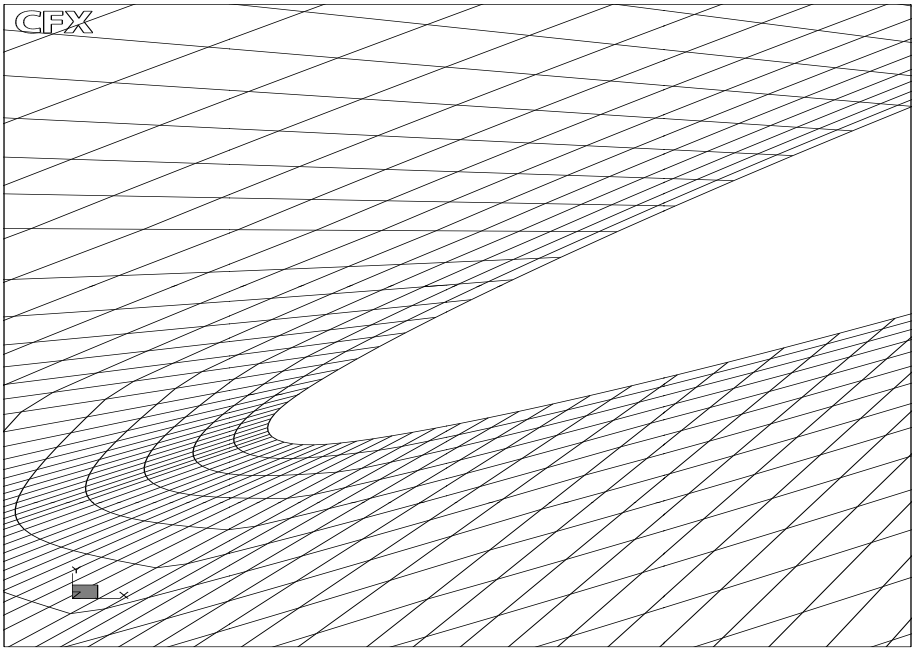

Figure 2.3: C-grid, leading edge

The grid is refined at the leading edge by adding more points on the profile (fig. 2.3). However, we notice that cells are higly skewed and not rectangular, affecting the flow resolution.

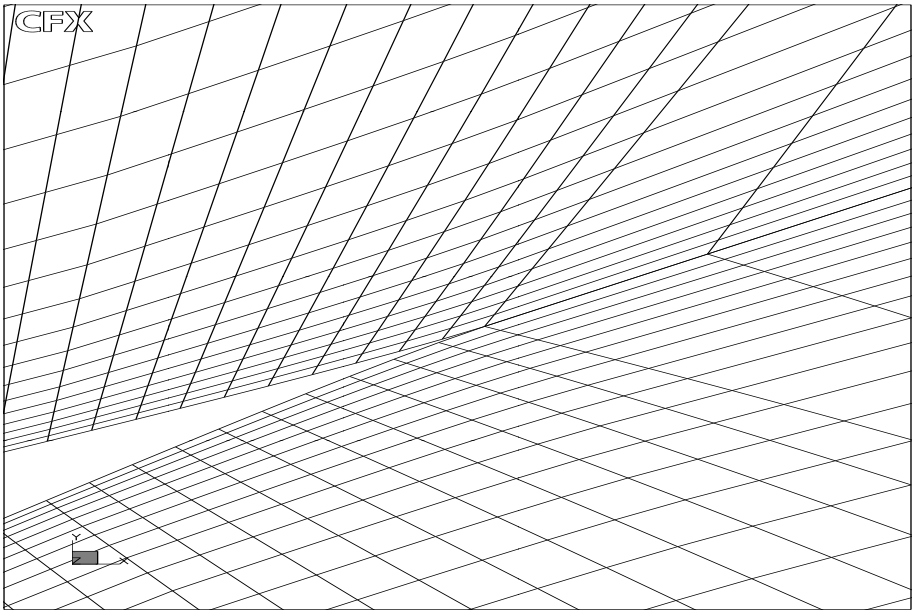

Figure 2.4: C-grid, trailing edge

The grid is refined at the trailing edge by adding more points on the profile too, but the node distribution behind the trailing edge is not the most appropriate, making big size differences between the neighbouring cells (fig. 2.4).

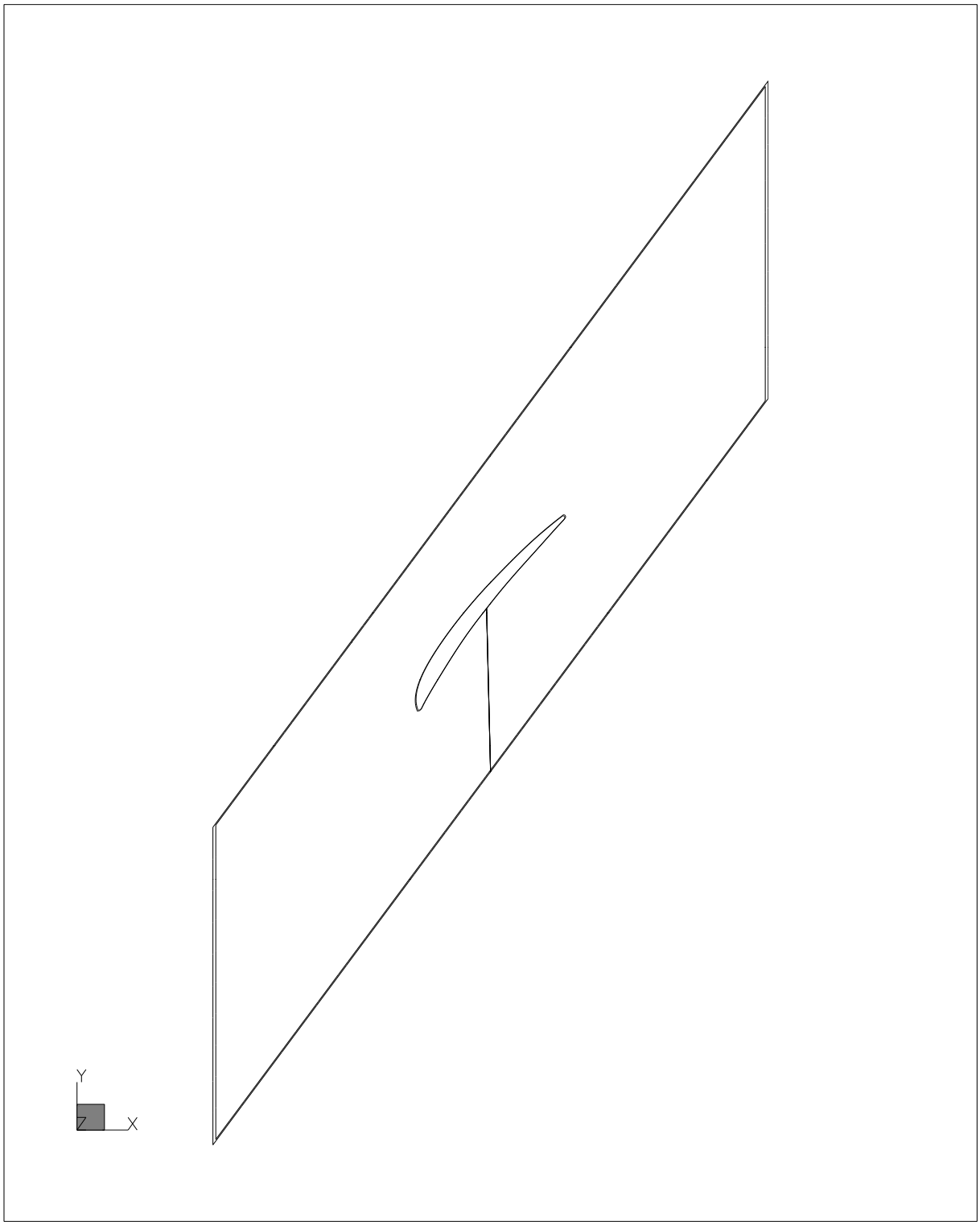

Figure 2.5: Topological lines for a O-grid

The O-grid is generated again with a template simple and easy to use, and it has the same characteristics as the previous C-grid, as shown in fig. 2.5.

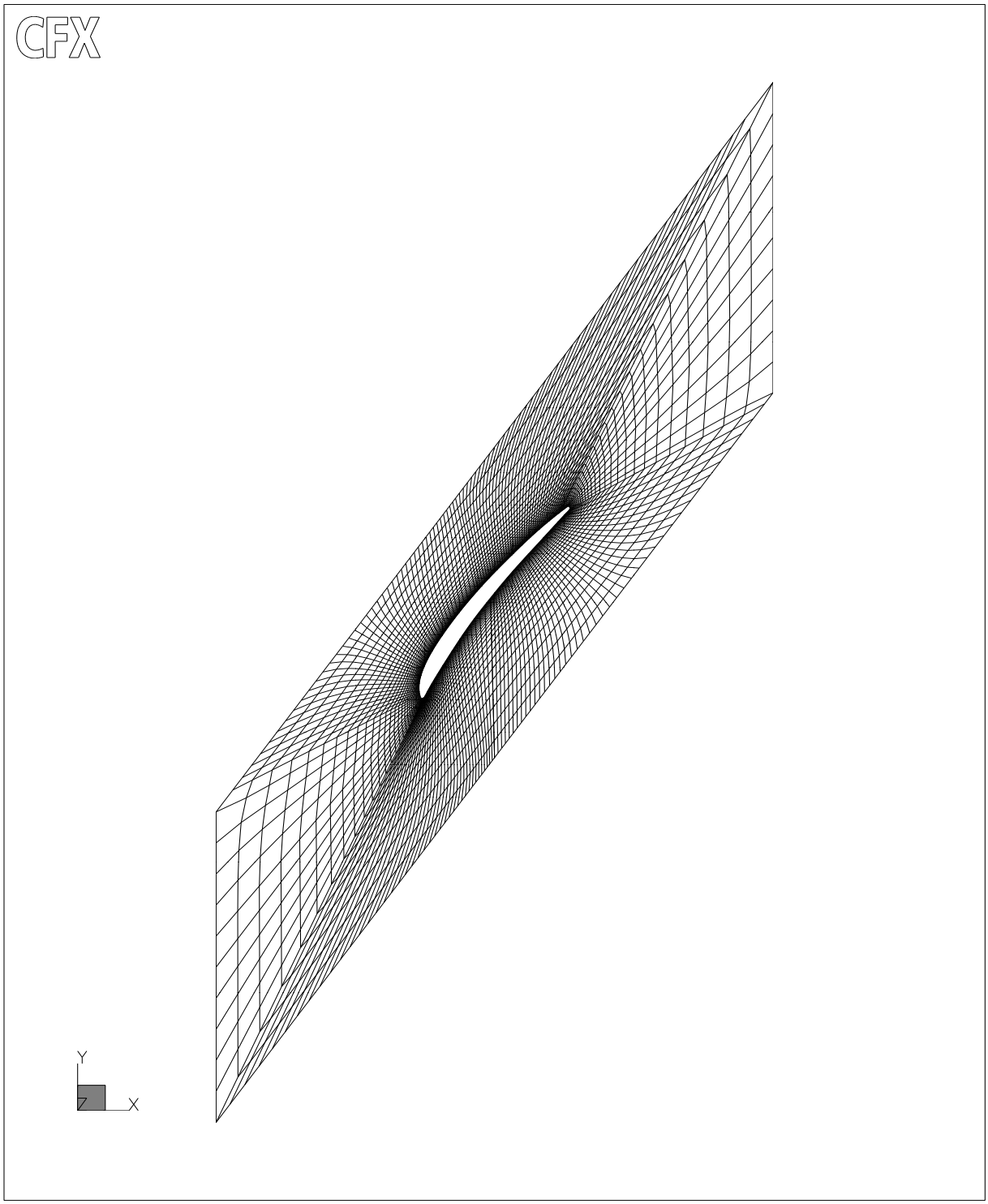

Figure 2.6: O-grid

This time the grid lines following the I-index are disposed in O around the profile, allowing a better discretization around the trailing edge. We notice that due to the topology and the fact that the inlet is far from the profile, the cells are highly skewed on diagonals.

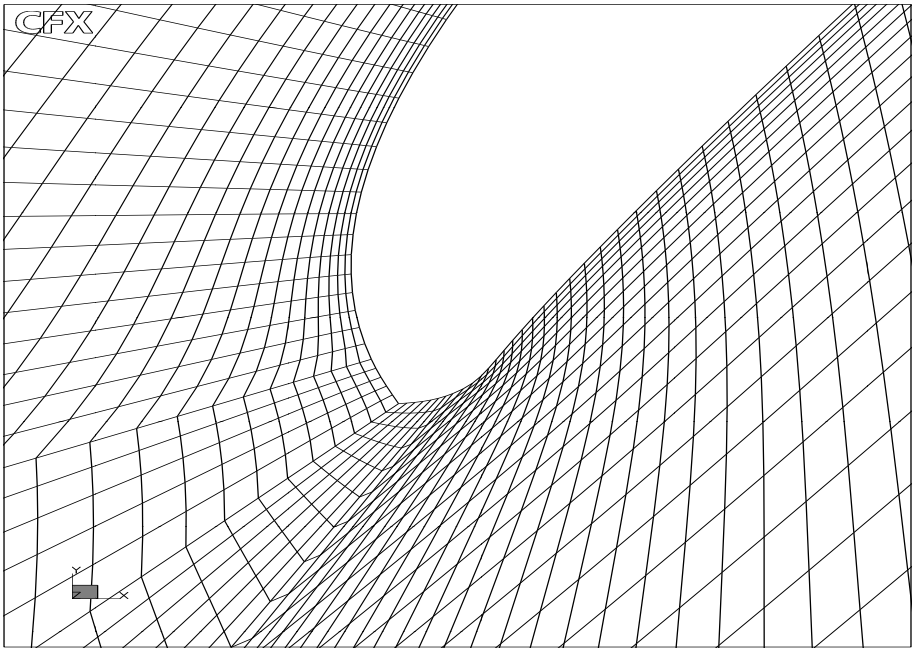

Figure 2.7: O-grid, leading edge

Altough the upper part is well discretized and the cells are rectangular, the grid on the lower surface of the leading edge is not very good. The cell shape becomes even worse for highly stagger angles.

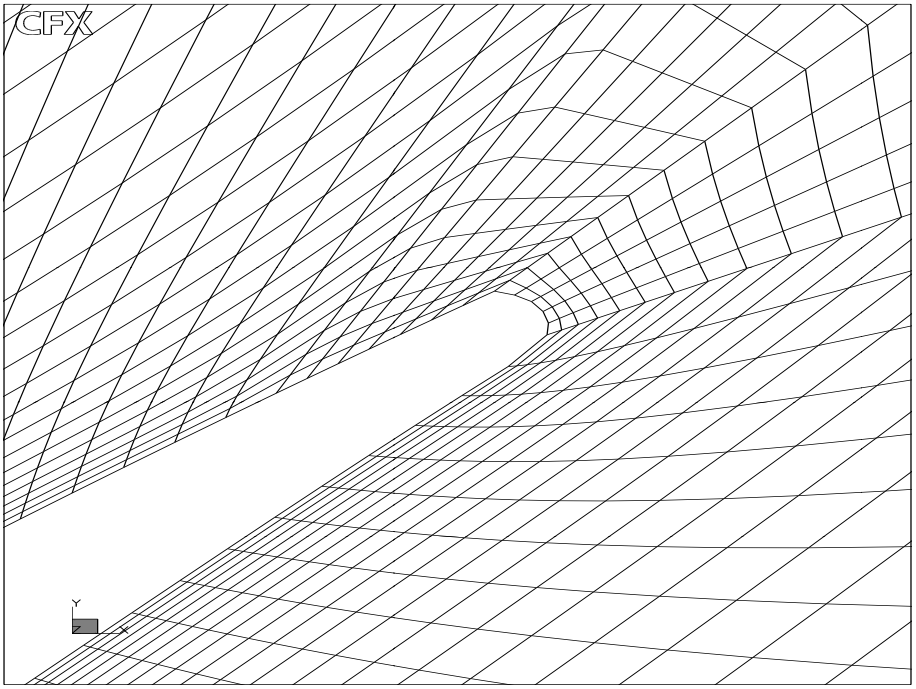

Figure 2.8: O-grid, trailing edge

The trailing edge is better discretized by this type of grid.

2.3.2 Multi-block grids with one block around the profile

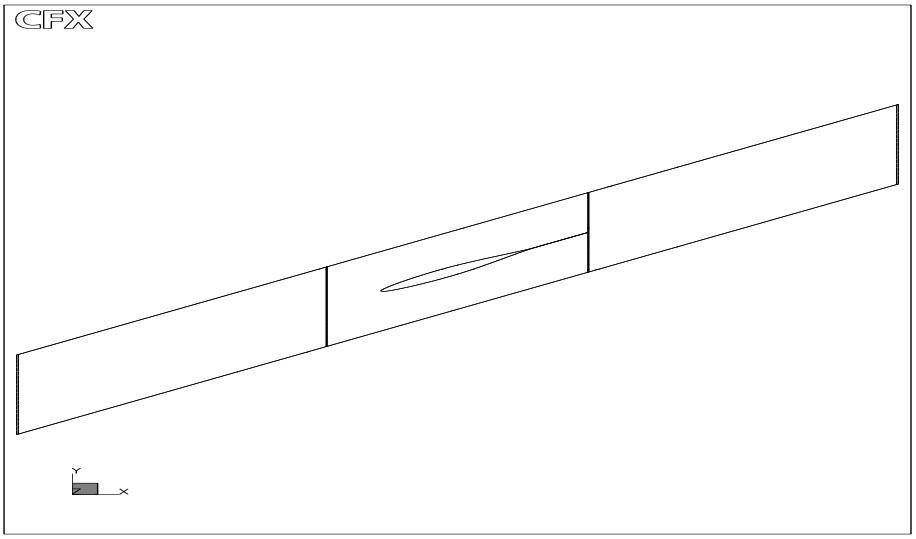

Figure 2.9: Topological lines for a multi-block grid around a C-block

This grid has the central block exactly the same as the previous C-grid, and 2 more blocks were added in the inlet and outlet regions. Thus, we can reduce the length of the central

block, allowing a better discretization of this region, and we can coarsen the grid at the inlet at outlet regions for an acceleration of flow resolution.

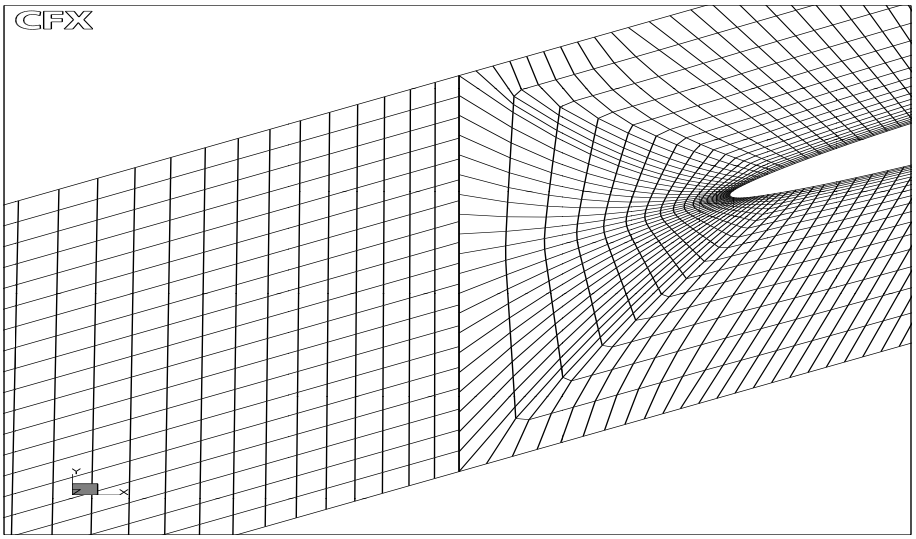

Figure 2.10: Blocks attachment around a C-block

We notice that the nodes of the two blocks do not necessarily coincide at the block interface. This is easily handled by TASCflow by specifying a GGI (General Grid Interface) attachment condition for this interface.

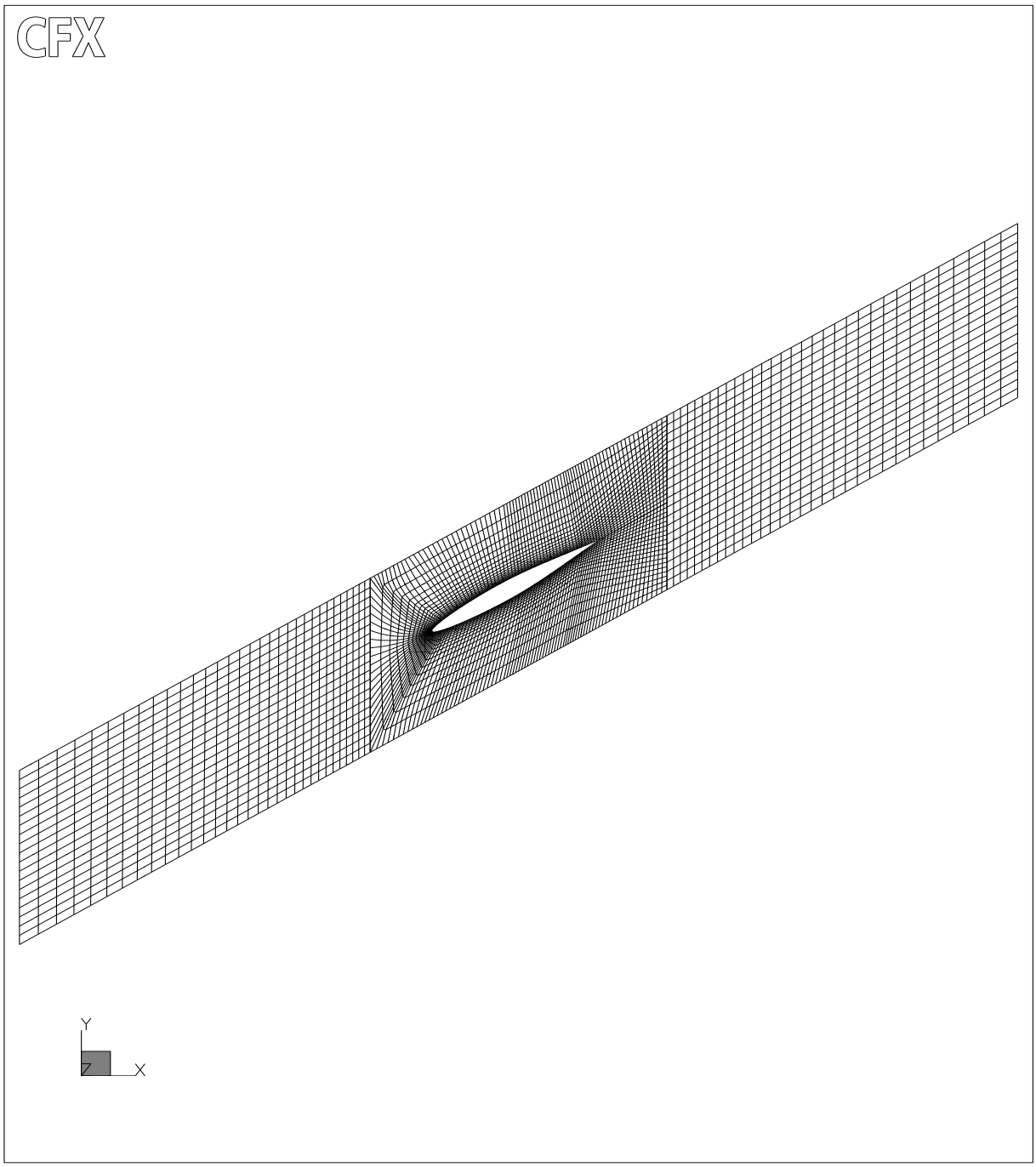

Figure 2.11: Multi-block grid around a C-block

This global view shows the grid quality in the central block. We see clearly that multi-block grid like this is better than the previous mono-block grid. The same improvement can be done for a O-grid by adding blocks around the central block.

2.3.3 Multi-block grids with several blocks around the profile

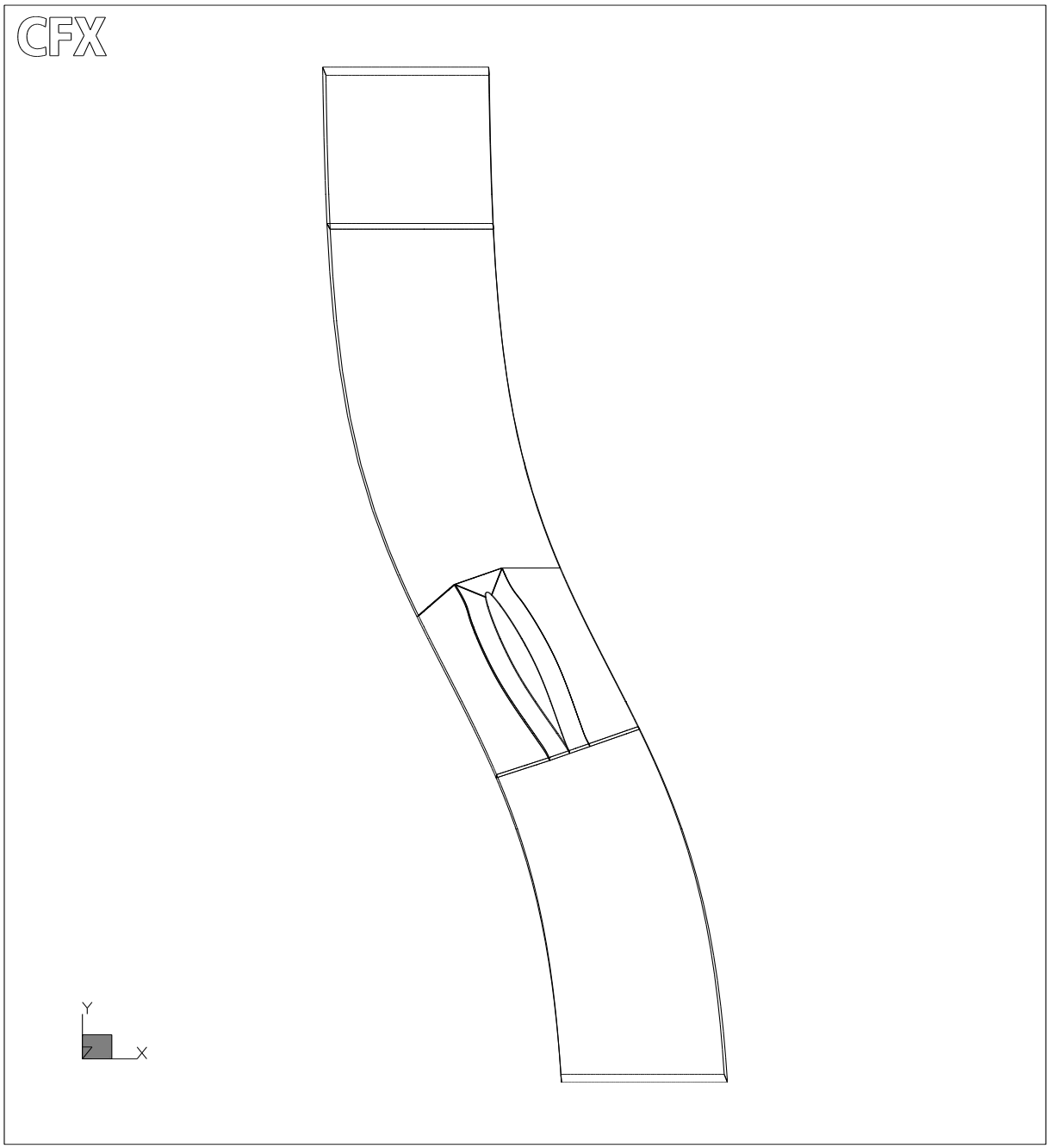

Figure 2.12: Topological lines for multi-block grid around a C-multiblock

This grid is realized by a second-generation template developed recently by VALEO TM with AEA Technology. This template can generate 2–D and 3–D grids and was designed to improve the grid around the profile and the boundary layer resolution. The second advantage is that in the optimization code, the mesh will be deformed only in the blocks around the profile, keeping the external border of these blocks unchanged.

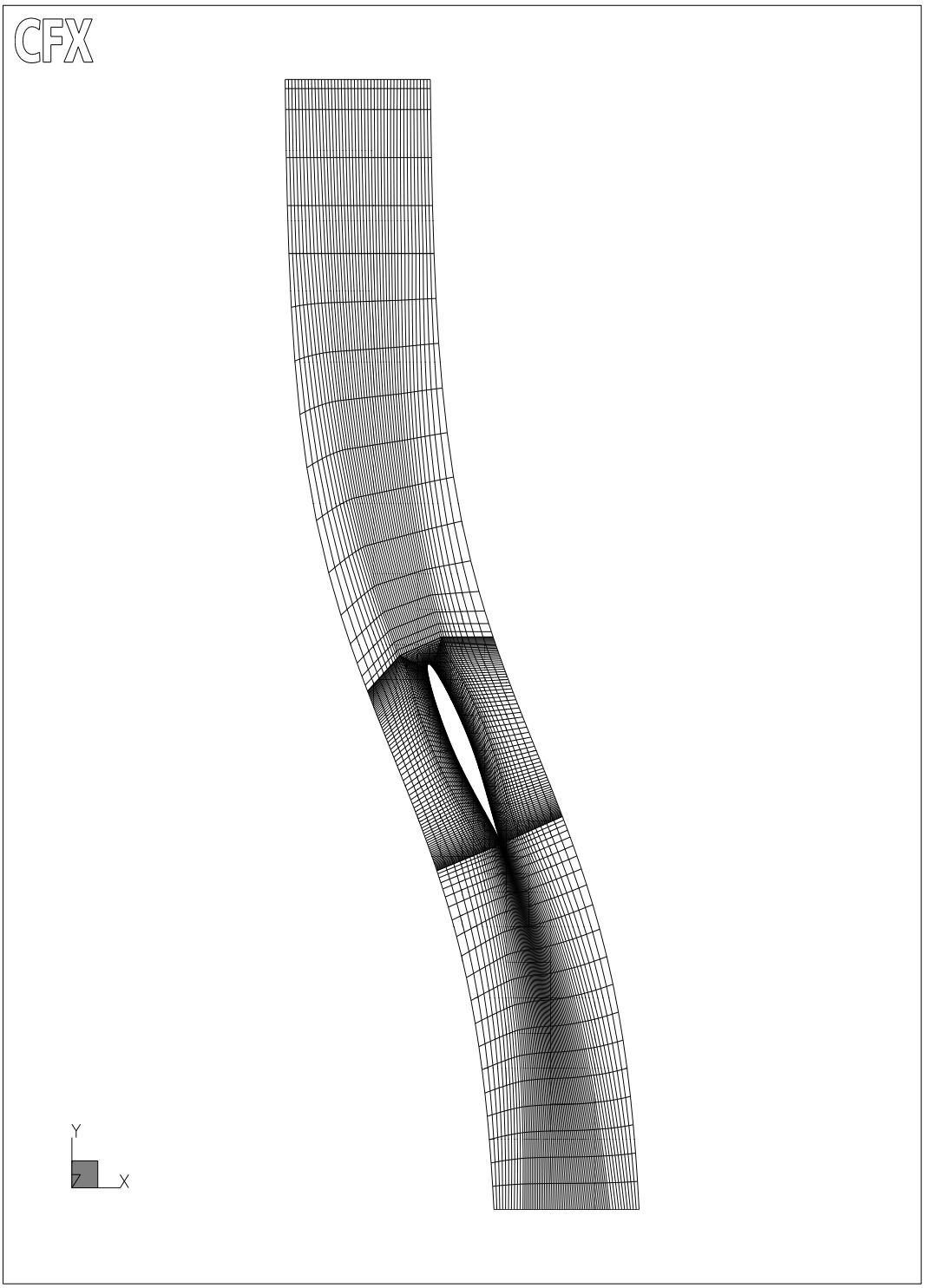

Figure 2.13: multi-block grid around a C-multiblock

We see that the C-grid of fig. 2.13 is well behaved around the profile. However, the grid being structured, we notice unnecessary refinements in different regions. At present the template is being improved by the AEA Technologies team.

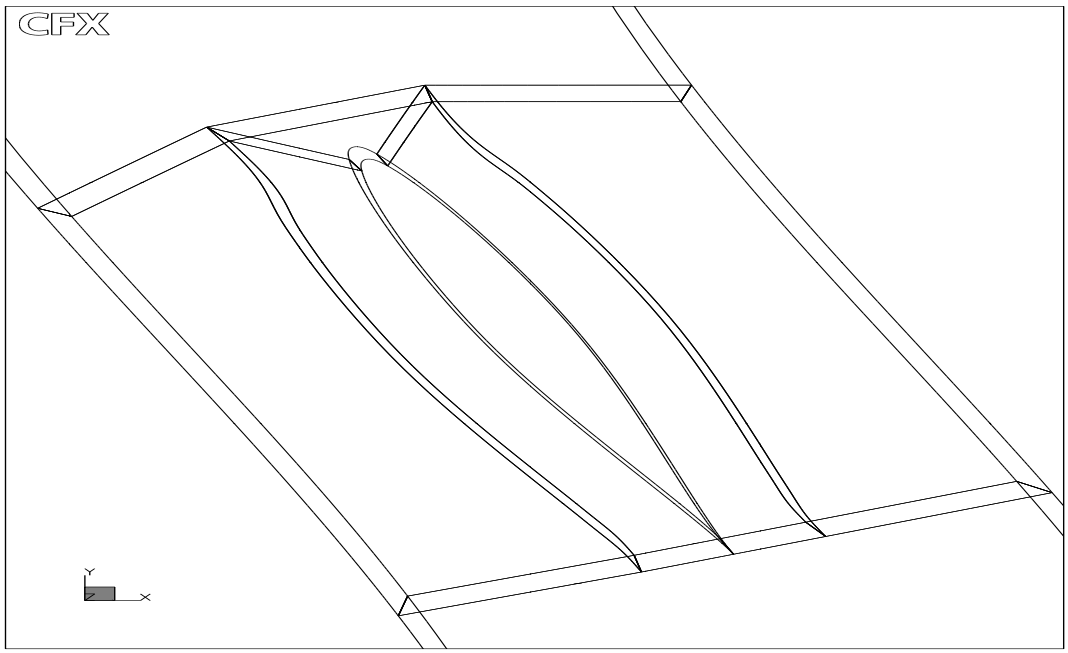

Figure 2.14: Topological lines for the C-multiblock

There are 3 blocks for the boundary layer modelization.

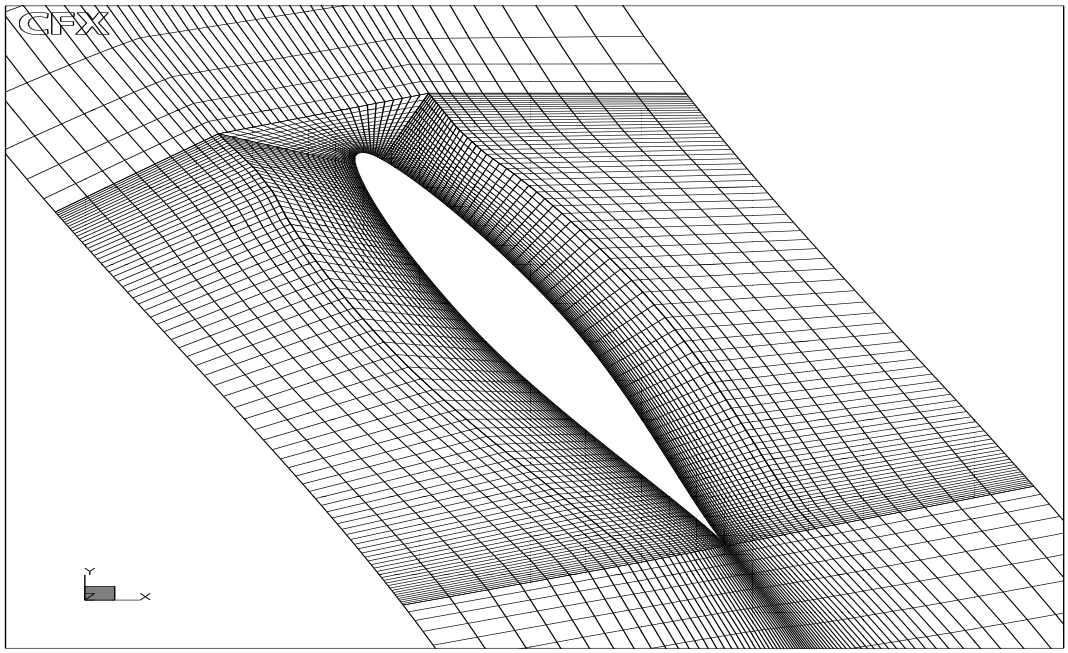

Figure 2.15: C-multiblock grid

The node distribution the blocks around the profile is realized with a geometrical ratio. Although the grid here is pretty good, there is still room for improvement, by allowing more flexibility to the topological lines, especially for highly cambered blade section.

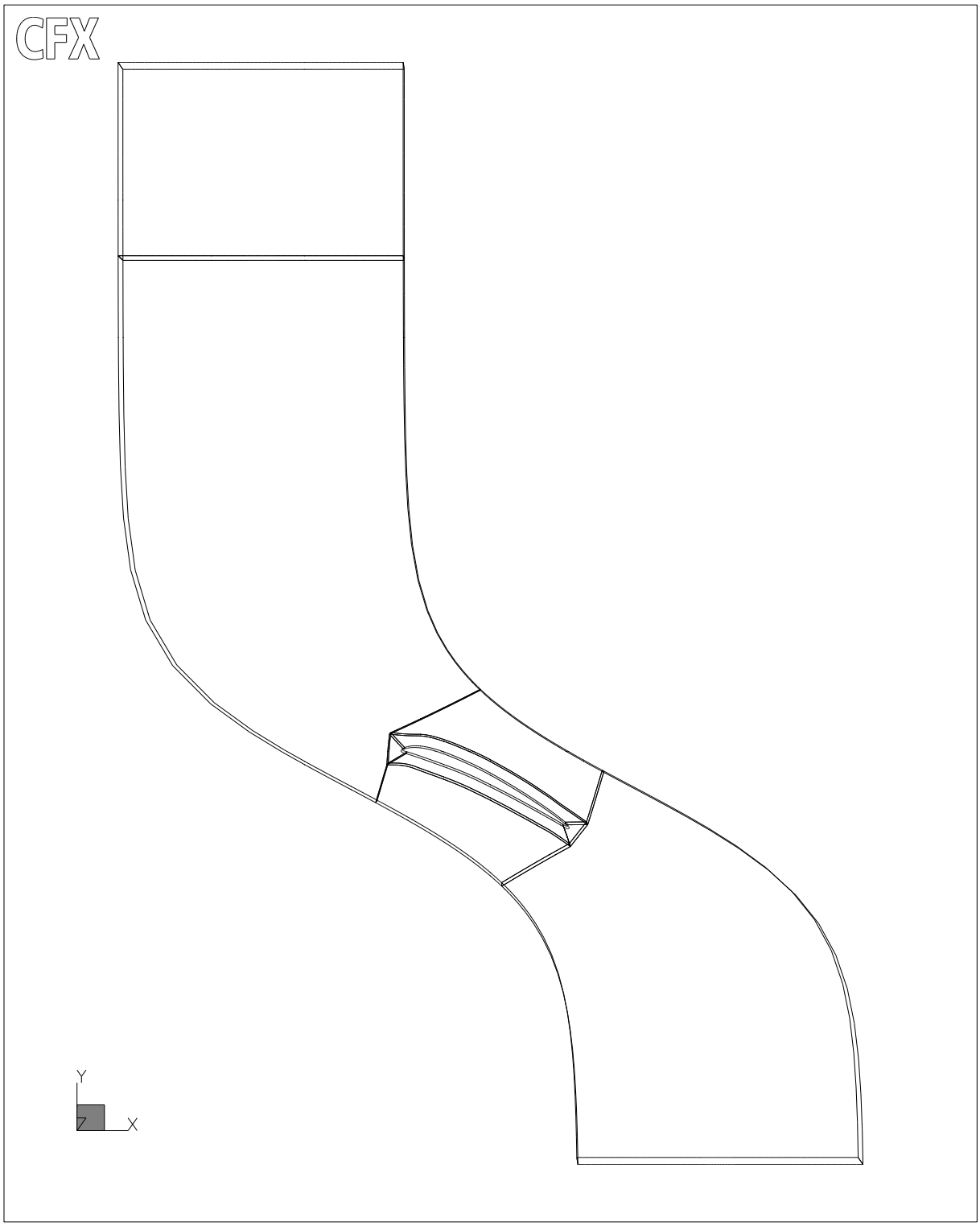

Figure 2.16: Topological lines for multi-block grid around a O-multiblock

This O-grid is achieved by the same type of template. The difference is that a new block is added at the trailing edge.

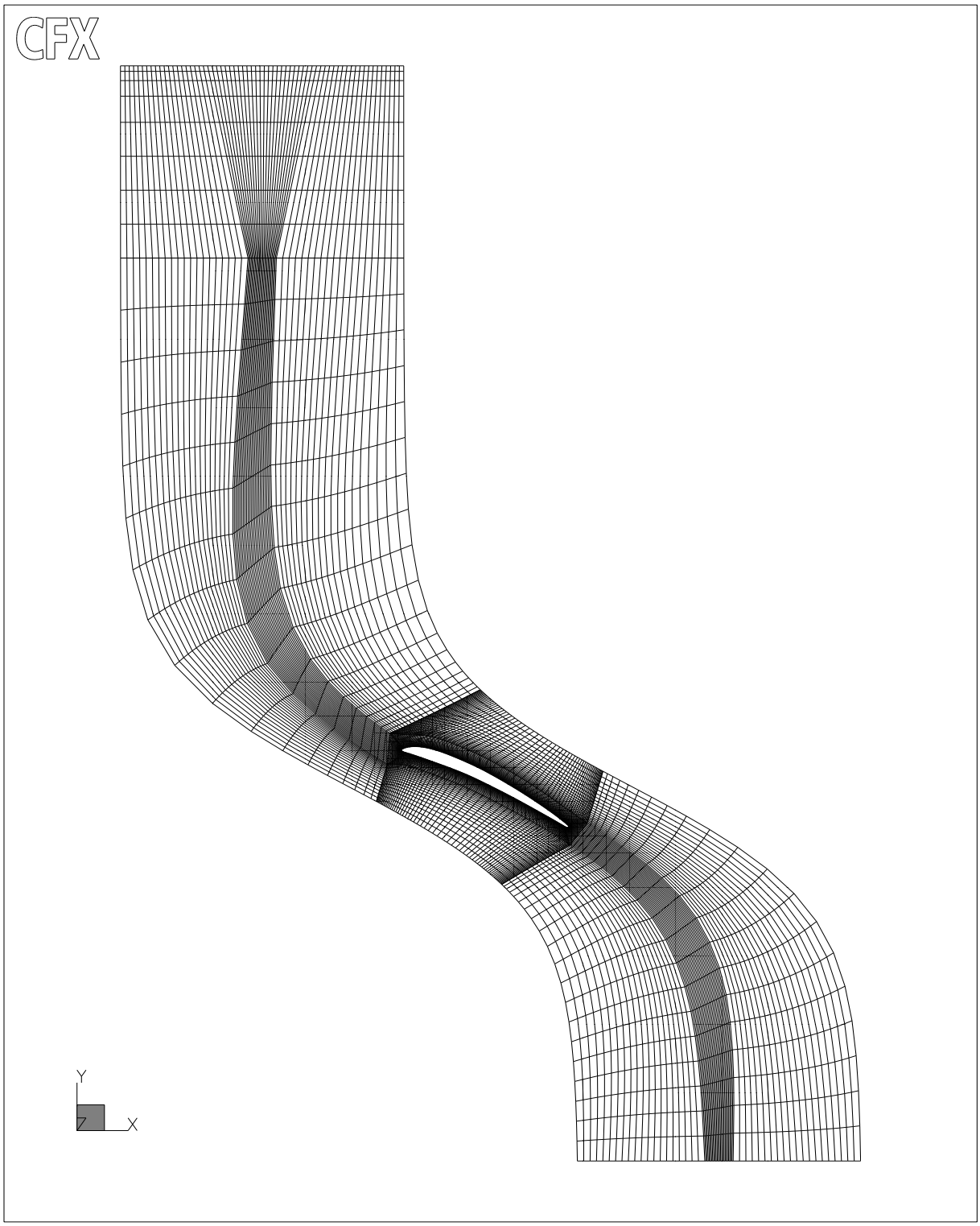

Figure 2.17: multi-block grid around a O-multiblock

We can see a lot of unnecessary refinements. This can be avoided by a better node distribution.

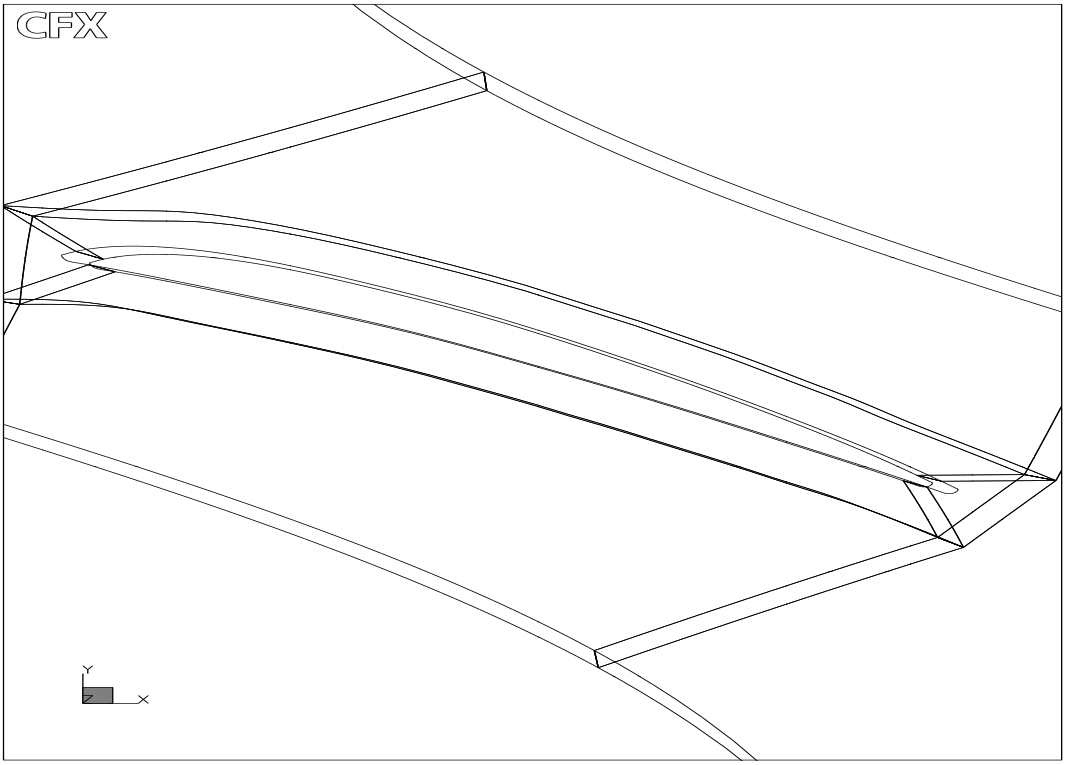

Figure 2.18: Topological lines for the O-multiblock

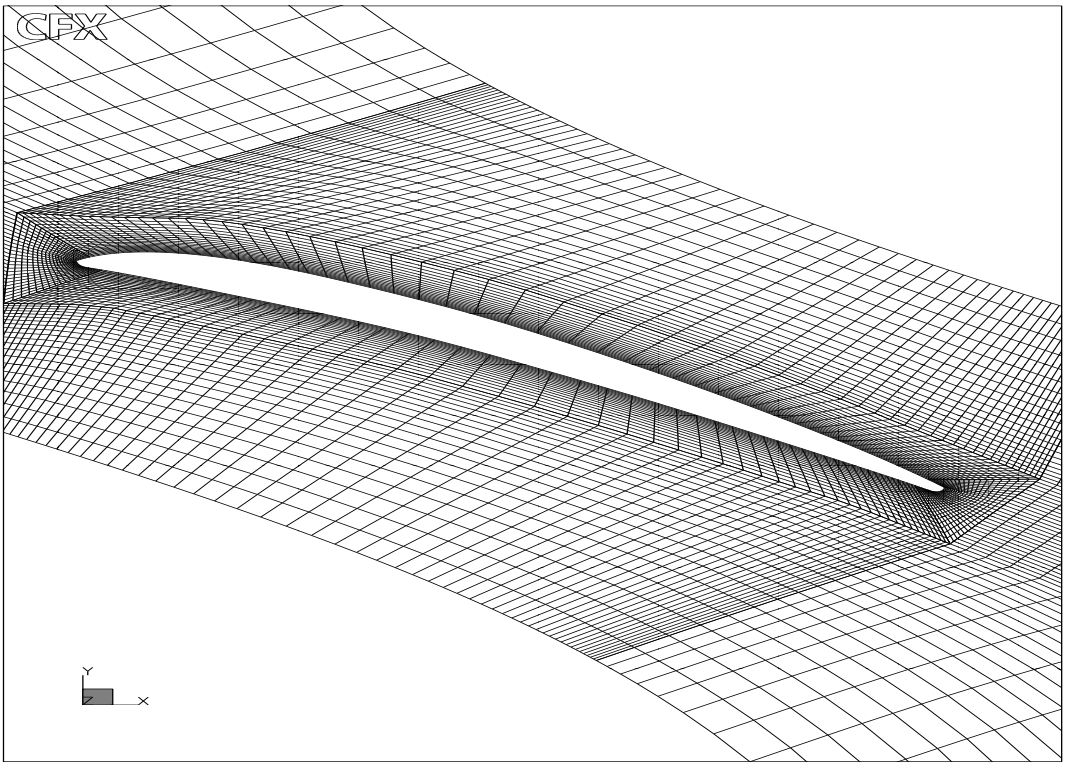

Figure 2.19: O-multiblock grid

Chapter 3

Optimization Code TASCOPT-3.0

3.1 Code description

3.1.1 Previous versions

The first version of the optimization code TASCOPT-1.0 used at VALEO TM released in June 1998 was the code OPTIM-2D from INRIA Rocquencourt slightly modified in order to use TASCflow structured grids. In fact in the optimization shell was added another program which transformed structured grids into unstructured grids of type *.amdba (and vice-versa). This was possible only for 2–D single-block grids in C. All the geometrical framework, as well as the optimization techniques were treated in this environment (geometry and flow data). Tests were performed for a NACA 65-010 profile.

The next version TASCOPT-2.0 was improved to handle with multi-block grids with one block around the profile, in C or O, and was released in January 1999. The same strategy of using an interface program between the optimization code and the solver was used. Two new optimization methods were added : the BFGS and the heavy-ball techniques. New tests for FC1001 and H383EC1 profiles were performed.

These first two versions were rather restrictive in terms of the grid handling. In fact, for strongly curved profiles the grids with a single block around the profile were quite poor, the flow resolution was not satisfactory and the mesh deformations were limited.

3.1.2 Actual version

The current version TASCOPT-3.0, released in June 1999, was completely reprogrammed for the TASCflow environment. This time the geometrical framework and the optimization procedures are treated for the specific grid and flow data of TASCflow.

The code was written in a modular manner, allowing easily to add, remove or change subroutines. Its structure is similar to the optimization algorithm presented in 1.3.6.

While the geometrical capabilities were greatly improved, from the point of view of the optimization methods only the steepest descent was implemented and other techniques like the BFGS method, the heavy-ball method or the Interior Point Algorithm are to be added in the future versions.

TASCOPT-3.0 consists of a shell called OPTIMTASC which links the optimization code with the flow solver and the output data generator. The structure of the shell is:

1. the flow is initialized from a previous computation;
2. optimization loops are performed, where the geometrical part and the optimization procedures are treated in optimtasc.exe and the flow resolution is done by tascflow3d;
3. the output data is generated by cost.exe.

3.1.3 Shell OPTIMTASC

The shell is:

```
#########################################################
#                    TASCOPT-3.0                        #
#                                                       #
#                by Mugurel STANCIU                     #
#          VALEO TM, INRIA Rocquencourt                 #
#                    June 1999                          #
#                 Shell OPTIMTASC                       #
#########################################################

TASC_UTILITY=/usr/tascflow/2.8/TASCflow/Utility
. $TASC_UTILITY/Ksetup

#########################################################
#     initial solution                                  #
#########################################################
echo 'Flow initialization'

rm -f *grd *gci *bcf *rso
rm -f shape.*
rm -f HISTORY1
touch HISTORY1
rm -f HISTORY2
touch HISTORY2
rm -f COST1
touch COST1
```

```
rm -f COST2
touch COST2
rm -f GRADIENT
touch GRADIENT

cp -f bcf_initial bcf
cp -f gci_initial gci
cp -f grd_initial grd
cp -f rso_initial rso
cp -f prm_initial prm
cp -f name.lun_initial name.lun

#########################################################
#       begin optimization loop                         #
#########################################################

niterations=30
iteration=0
while [ $iteration -le $niterations ]
do
iteration=‘expr $iteration + 1‘

echo $iteration > contor
echo ’Optimization loop: iteration ’‘expr $iteration‘

#########################################################
#     optimization                                      #
#########################################################
echo ’New shape computation’

optimtasc.exe

#########################################################
#       output data                                     #
#########################################################

cat HISTORY1 hist1.data >> HISTORYJ
mv HISTORYJ HISTORY1
cat HISTORY2 hist2.data >> HISTORYJJ
mv HISTORYJJ HISTORY2
cat GRADIENT gradmod >> GRADIENT2
mv GRADIENT2 GRADIENT

cost.exe
```

```
#######################################################
#      save data                                      #
#######################################################
echo 'Save new shape in shape.'`expr $iteration`

cp -f blade.out shape.$iteration

#######################################################
#     flow resolution for the new profile             #
#######################################################
echo 'Flow computation'

tascflow3d -s10

#######################################################
#      end optimization loop                          #
#######################################################

done
```

The converged initial solution, the initial grid and the other files used in flow resolution *_initial are copied into "work" files which will be modified during the optimization process. The number of optimization steps is defined in the parameter **niterations**. The optimization is performed by optimtasc.exe and the flow resolution by tascflow3d. The memory necessary for the flow compution is defined by the parameter -s10 (see TASCflow User Documentation, Ref. [1]). The program cost.exe generates files for postprocessing with the cost function and the gradient convergence during the optimization. The shapes for each optimization step are saved in shape.*.

3.1.4 Optimization code optimtasc.exe

The structure of the code optimtasc.exe is:

- The main program is *optim_tasc*. The global variables are read in the file p.h. This program corresponds to an optimization iteration where the old grid, the profile and the solution are read, the gradient is computed, a gradient method is used for obtaining the new profile, a smoothing is performed for this profile, the old grid is deformed to match to the new profile and the new grid is saved.
  - the grid geometry is read from DATA_OPTIM_TASC;
  - the TASCflow solution and grid are read with specific TASCflow Database Interface subroutines: *tginit_test* (initialize the Database), *trread* (read the

Database into memory), *trngbl* (retrieve the number of subgrid blocks), *trgdim* (find dimensions of the grid), *trscal* (extract scalar fields from the Database, such as nodes coordinates, pressure, velocities and turbulent viscosity).

- the geometrical and the flow data for the shape to be optimized is gathered in *optim_gather_tasc* (data from each subgrid block containing the shape is centralized into global data corresponding to the shape and the surrounding blocks);
- the initializations are performed in *optim_init_tasc*. Optimization parameters, constraints and cost function parameters are read from DATA_OPTIM_TASC;
- the segments and the quadrangles on the shape surface are computed in *optim_segment_tasc*;
- the aerodynamical coefficients, the geometrical quantities of the shape and the cost function are computed in *optim_cost_tasc*;
- the gradient is computed in *optim_costad_tasc*. This subroutine is obtained by automatic differentiation of the subroutine *optim_cost_tasc* using the AD tool Odyssée from INRIA;
- the new shape is computed using a gradient method;
- the geometrical constraints are taken into account in *optim_project_tasc*;
- the new shape is smoothed in *optim_smooth_tasc*;
- each block around the shape is deformed to fit to the new shape in *optim_mesh_tasc*;
- the new grid is saved using TASCflow Database subroutines: *twngbl_test* (define the number of subgrid blocks), *twsscl_test* (write scalar fields to the Database, e.g. the nodes coordinates), *tgclose* (close the Database).

The code is compiled with the command gmake or make, using the following makefile:

```
LIBDIR = $(TASC_DIR)/Lib
LDLIBS = $(TASC_MISC)/Lib
TDI_A    = $(LIBDIR)/tdi.a

FC = f77
FFLAGS = -O -32 -Olimit 1124

F_SRC=optim_tasc.f optim_cost_tasc.f optim_costad_tasc.f     \
optim_gather_tasc.f optim_init_tasc.f optim_segment_tasc.f   \
optim_project_tasc.f optim_smooth_tasc.f optim_mesh_tasc.f   \
optim_pres_tasc.f  twsscl_test.f gtgrd3_test.f twngbl_test.f \
tginit_test.f optim_test0.f optim_test1.f optim_test2.f      \
optim_test3.f  optim_test4.f optim_test5.f
OBJ=$(patsubst %.f,%.o,$(F_SRC))
```

```
ARCHIVES = $(LIBDIR)/bob3d.a $(LIBDIR)/system.a $(LIBDIR)/aclgraph.a \
       $(LIBDIR)/tasclib.a $(LDLIBS)/libz.a $(LIBDIR)/system.a

optimtasc.exe: $(OBJ)
$(FC) $(FFLAGS) -o $@ $(OBJ) $(TDI_A) $(ARCHIVES)
```

3.2 Input and output files

3.2.1 Input file

The input file DATA_OPTIM_TASC is:

```
--- Grid blocks ------------------------------------------------------
4                       -> nblocks describing the shape
6 35  1  6  6  1  3     -> block number, i1, i2, j1, j2, k1, k2
5  6  6  1  6  1  3     -> block number, i1, i2, j1, j2, k1, k2
7  1 35  1  1  1  3     -> block number, i1, i2, j1, j2, k1, k2
8  1  1  6  1  1  3     -> block number, i1, i2, j1, j2, k1, k2
--- Local - global grid correspondance -------------------------------
--- Block 6 ----------------------------------------------------------
 0     1     0     0
88     0     1     0
 0     0     0     1
--- Block 5 ----------------------------------------------------------
35     1     0     0
92     0     1     0
 0     0     0     1
--- Block 7 ----------------------------------------------------------
47     0    -1     0
36    -1     0     0
 4     0     0    -1
--- Block 8 ----------------------------------------------------------
40     1     0     0
35     0     1     0
 0     0     0     1
--- Pressure difference ----------------------------------------------
9  1  1  1  46  1  3   -> block number inlet,  i1, i2, j1, j2, k1, k2
4 21 21  1  46  1  3   -> block number outlet, i1, i2, j1, j2, k1, k2
--- Optimization parameters ------------------------------------------
0                       -> ibp (-2 sdn -1 bfgs, 0 sd) for method
0.    0.                -> dxnn, dynn (initial velocity for heavyball)
```

```
1.e-3 0.                -> ro(steepest descent), robp(heavyball)
0.001  30               -> xttt, nsmo for smoothing
2.                      -> beta for mesh deformation
--- Geometrical constraints ----------------------------------------
0.1  0.001 0.1  0.001  -> xlim (%, max), ylim (%, max) for deformation
0.250  .305             -> xmin,xmax
-0.035  1.055           -> ymin,ymax
-10.   10.              -> zmin,zmax
--- Aerodynamic coefficients ---------------------------------------
-90. 0.                 -> tetadeg1,tetadeg2 for Cd, Cl
0.0 0.0                 -> xelast, yelast for Cm
--- Cost function --------------------------------------------------
0.                      -> xjam(1)   coef  Pres (inverse design)
0.                      -> xjam(2)   Cd0
0.                      -> xjam(3)   Cl0
0.                      -> xjam(4)   Cm0
0.                      -> xjam(5)   Cx0
0.                      -> xjam(6)   Cy0
0.                      -> xjam(7)   Cz0
0.                      -> xjam(8)   Mx0
0.                      -> xjam(9)   My0
0.                      -> xjam(10)  Mz0
0.                      -> xjam(11)  Vol0
0.                      -> xjam(12)  coef |Cd-Cd0|
0.                      -> xjam(13)  coef  |1/Cd|
0.                      -> xjam(14)  coef |Cl-Cl0|
0.                      -> xjam(15)  coef  |1/Cl|
0.                      -> xjam(16)  coef |Cm-Cm0|
0.                      -> xjam(17)  coef  |1/Cm|
0.                      -> xjam(18)  coef |Cx-Cx0|
0.                      -> xjam(19)  coef  |1/Cx|
0.                      -> xjam(20)  coef |Cy-Cy0|
0.                      -> xjam(21)  coef  |1/Cy|
0.                      -> xjam(22)  coef |Cz-Cz0|
0.                      -> xjam(23)  coef  |1/Cz|
0.                      -> xjam(24)  coef |Mx-Mx0|
0.                      -> xjam(25)  coef  |1/Mx|
0.                      -> xjam(26)  coef |My-My0|
0.                      -> xjam(27)  coef  |1/My|
0.                      -> xjam(28)  coef |Mz-Mz0|
0.                      -> xjam(29)  coef  |1/Mz|
1.                      -> xjam(30)  coef |Vol-Vol0|
0.                      -> xjam(31)  coef  |1/Vol|
0.                      -> xjam(32)  coef  |Cx/Cy|
0.                      -> xjam(33)  coef  |Cd/Cx|
```

```
0.                    -> xjam(34)  coef  |Cd/Cl|
0.                    -> xjam(35)  coef  |Cl/Cd|
-------------------------------------------------------------------
```

This file contains all the initialization data for the grid geometry, the optimization method, the constraints and the cost function. TASCtool should be used to see the grid, the number of blocks and their parameters before editing the DATA_OPTIM_TASC file. The lines with comments separating different groups of data in this file must not be erased.

The structure of DATA_OPTIM_TASC is:

- comment line.
- **nblocks** represents the number of subgrid blocks containing the profile to be optimized. If the grid is monoblock **nblocks** can be 0 or 1.
- for each subgrid block containing the profile, the user must introduce the **block number** (as defined in TASCgrid) and the indexes defining the profile for this block: **i1**, **i2**, **j1**, **j2**, **k1**, **k2** (viewed in TASCtool. It is important that the blocks have to be introduced in DATA_OPTIM_TASC in a clock-wise order: the block containing the lower part of the profile, the block containing the leading edge, the block containing the upper part of the profile, and the block containing the trailing edge (if the mesh is in O).
- comment line.
- for each subgrid block containing the profile, after a comment line, the user must introduce the correspondance between the local coordinates of the block and the global coordinates of the grid. This can be found in the bcf file, after the characters **$$$LTG**, where for each block of the grid the grid dimensions (first column) and the correspondance between the local and the global information is defined with a translation vector (the second column) and a rotation matrix (the third, fourth and fifth columns) for the i, j, k indexes:

```
$$$LTG
    9 ! number of local grids defined by grid embedding / attaching.
    1 MAIN                      ! grid #, local grid name
   19     0     1     0     0  ! local / global information
   46     0     0     1     0  ! local / global information
    3     0     0     0     1  ! local / global information
```

  DATA_OPTIM_TASC needs only the translation vector and the rotation matrix, therefore the user must copy the colums 2-4 in the file.

- comment line.

- the **block number** containing the inlet region and the indexes defining this region: **i1**, **i2**, **j1**, **j2**, **k1**, **k2**. This is needed for the computation of the inlet pressure.
- the **block number** containing the outlet region and the indexes defining this region: **i1**, **i2**, **j1**, **j2**, **k1**, **k2**. This is needed for the computation of the outlet pressure.
- comment line.
- **ibp** defines the optimization method. **ibp=0** coresponds to the steepest descent method, **ibp=-1** to the BFGS method (not currently implemented) and **ibp=-2** to the steepest descent method from the BFGS simplified method (similar to the standard steepest descent method).
- **dxnn** and **dynn** represent the initial velocity components for the heavy-ball method.
- **ro** represents the optimization step size and **robp** the step size for the heavy-ball method. If the heavy-ball method is not used then **robp=0**. Usually **robp** should be an order of magnitude less than **ro**. The stepsize **ro** defines the profile changement (if greater, the profile changes more and the intermediate flow solving should be longer) - see 1.20.
- **xttt** characterises the smoothing operator and **nsmo** defines the number of Jacobi iterations performing the smoothing. The nodes with the relative distance to the medium curve defined by the adjacents points greater than **xttt** are smoothed. The greater **nsmo** is, the smoother the profile will be (for **nsmo=0** no smoothing is performed) - see 1.12.
- **beta** defines the node deformation propagation in the grid. If greater, the grid will be deformed only in the vicinity of the deformed shape - see 1.13.
- comment line.
- **xlim** percentual and maximal and **ylim** percentual and maximal define the maximal deformation allowed for the profile. The percentual parameter represents the percent of the local coordinate x or y and the maximal represents the absolute value of the maximal deformation. The projection operator takes into account the minimum between these two values.
- **xmin**, **xmax**, **ymin**, **ymax**, **zmin**, **zmax** define the box of admissible space of change for the profile. The nodes having the coordinates less than the minimal value and greater than the maximal value are kept unchanged.
- comment line.
- **tetadeg1** and **tetadeg2** define the angle in the horizontal plane and the vertical plane for the aerodynamic coefficients C_d and C_l computation.
- **xelast** and **yelast** define the reference point for the C_m coefficient calculation.
- comment line.

- **xjam(1)** defines if a direct design (**xjam(1)=0**) or an inverse design (**xjam(1)=1**) is performed. In the last case a file with the target pressure is needed (not currently implemented).
- **xjam(2)** to **xjam(11)** represent the initial values for the aerodynamic coefficients and the volume of the shape, needed in the cost function definition. If such a value is 0, a cost function like $J = |C_d - C_d^0|$ will perform a C_d minimization. State constraints can be imposed by adding to the cost function a term like $|V - V^0|$, this V^0 representing the real initial volume (and not zero).
- **xjam(12)** to **xjam(36)** represent the coefficients in front of each term in the definition of the cost function: $J = \alpha|C_d - C_d^0|$. If such a value is 0, the corresponding term is not taken into account in the cost function definition.
- comment line.

3.2.2 Output files

The output files are: HISTORY1, HISTORY2, COST1, COST2, GRADIENT, shape.*.

The file HISTORY1 contains the history during the optimization loop for the aerodynamic coefficients C_x, C_y, C_z and for the momentum coefficients M_x, M_y, M_z. The file has 6 columns, each for a coefficient, and each line corresponds to an optimization iteration:

```
-0.11235E-01  0.14640E-01 -0.94162E-10  0.36601E-03  0.28087E-03 -0.38072E-02
-0.11346E-01  0.14707E-01  0.14089E-07  0.36767E-03  0.28364E-03 -0.37661E-02
-0.11417E-01  0.14780E-01  0.44354E-08  0.36949E-03  0.28543E-03 -0.37330E-02
-0.11470E-01  0.14834E-01  0.49722E-08  0.37085E-03  0.28676E-03 -0.37002E-02
-0.11517E-01  0.14879E-01  0.28072E-08  0.37196E-03  0.28791E-03 -0.36777E-02
-0.11563E-01  0.14925E-01 -0.19915E-08  0.37312E-03  0.28906E-03 -0.36716E-02
-0.11599E-01  0.14959E-01 -0.84709E-09  0.37397E-03  0.28996E-03 -0.36737E-02
-0.11627E-01  0.14988E-01 -0.30386E-09  0.37469E-03  0.29067E-03 -0.36865E-02
-0.11645E-01  0.15007E-01 -0.59484E-09  0.37519E-03  0.29114E-03 -0.37041E-02
-0.11656E-01  0.15020E-01 -0.41473E-09  0.37549E-03  0.29140E-03 -0.37239E-02
```

The file HISTORY2 has the same structure and contains the history of the cost function J, the inlet-outlet pressure difference $\triangle p$, aerodynamic coefficients C_d, C_l, C_m and the shape volume V.

The files COST1 and COST2 contain the same data as HISTORY1, respectively HISTORY2, but this time the values are adimensionalized with respect to the initial state values (before optimization), allowing a quick look at the algorithm efficiency in percentage. For example, the files COST2 looks like:

```
1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
```

```
0.980321    1.010112    0.987002    1.006832    0.989205    1.013653
0.975382    1.016300    0.987500    1.012406    0.980511    1.027183
0.971583    1.020820    0.987667    1.016560    0.971895    1.039268
0.967784    1.024805    0.987210    1.020058    0.965986    1.049262
0.964516    1.028069    0.987293    1.023610    0.964383    1.056581
0.961325    1.030716    0.986670    1.026343    0.964935    1.062700
0.959426    1.032466    0.986836    1.028529    0.968297    1.069311
0.958438    1.033187    0.987210    1.030005    0.972920    1.076599
0.958134    1.033187    0.987750    1.030934    0.978120    1.084379
```

The file GRADIENT contains the history of the gradient module during the optimization loop. The first column corresponds to the gradient module as it is calculated by the subroutine *optim_costad_tasc*, the second one to the module of the gradient after projection and the third one to the module of the gradient after smoothing. This file shows the convergence of the gradient.

The file shape.* corresponds to the shape at iteration * and contains the coordinates x, y and z of the nodes of the shape. TASCOPT-3.0 generates as many shape.* files as optimization iterations are performed, allowing comparisons of the different shapes obtained during the optimization.

Chapter 4

Numerical Results

4.1 NACA 65-010

The initial data for this case are:

- initial profile NACA 65-010;
- incompressible turbulent flow, $Re = 4.5 \cdot 10^5$, $M = 0.1$;
- angle of incidence: $\alpha = 12^o$, stagger angle $\beta = 45^o$;
- solidity: $\sigma = 1.0$;
- number of control parameters on the shape x_c: $n_c = 140$;
- local geometrical constraints: fixed points on the leading and trailing edges;
- maximal distortion allowed for the shape 0.008;
- step size 0.04;
- cost function $J = C_d/C_l$;
- optimization iterations: 25;
- progressive optimizations: the intermediate solutions were considered converged when the residual decreased one order of magnitude.

The optimization results are:

- the optimization process cost was approximately 1.2 works (1 work $\equiv$ 1 converged flow analysis);
- the cost function $J = C_d/C_l$ was reduced by 4.5 %;

- the inlet-oulet pressure difference was increased by 1.5 %;
- the shape volume was increased by 1.7 %.

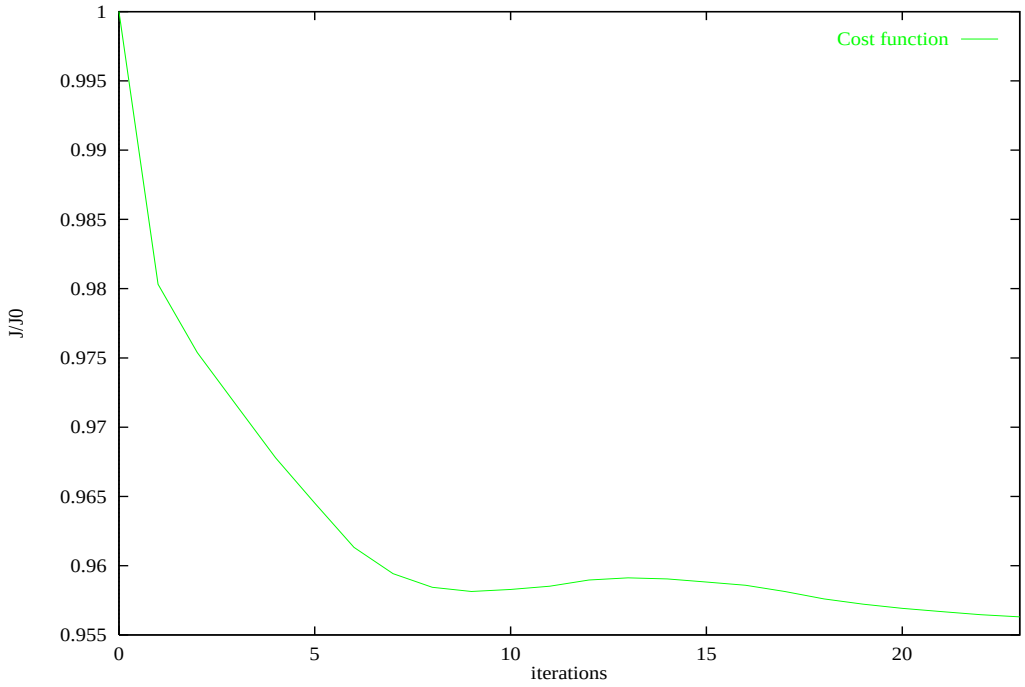

Figure 4.1: Cost function history

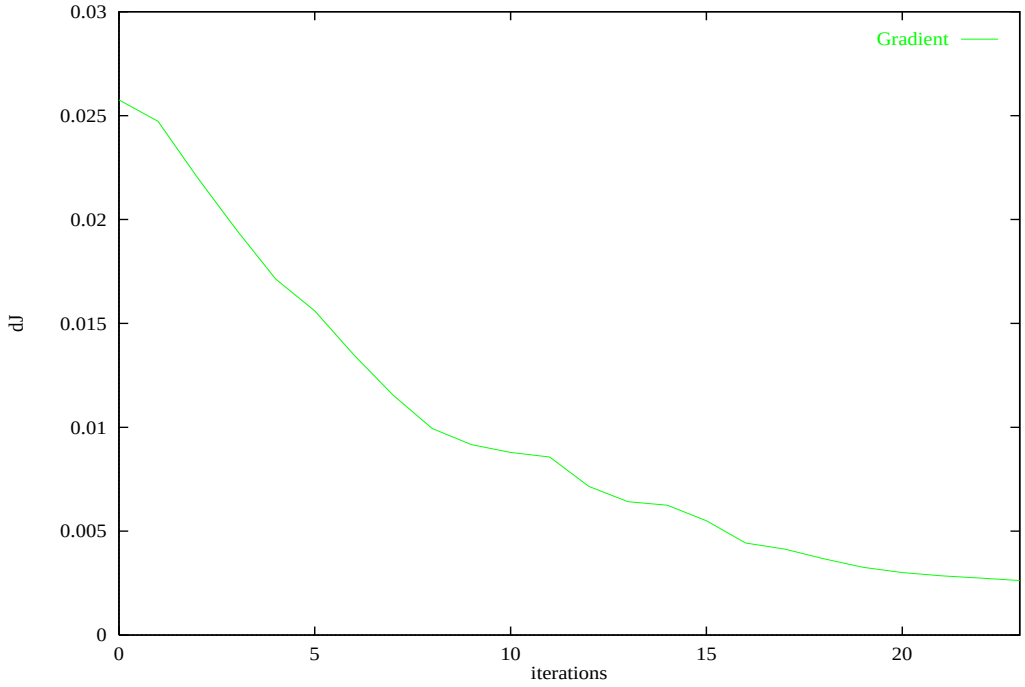

Figure 4.2: Gradient convergence

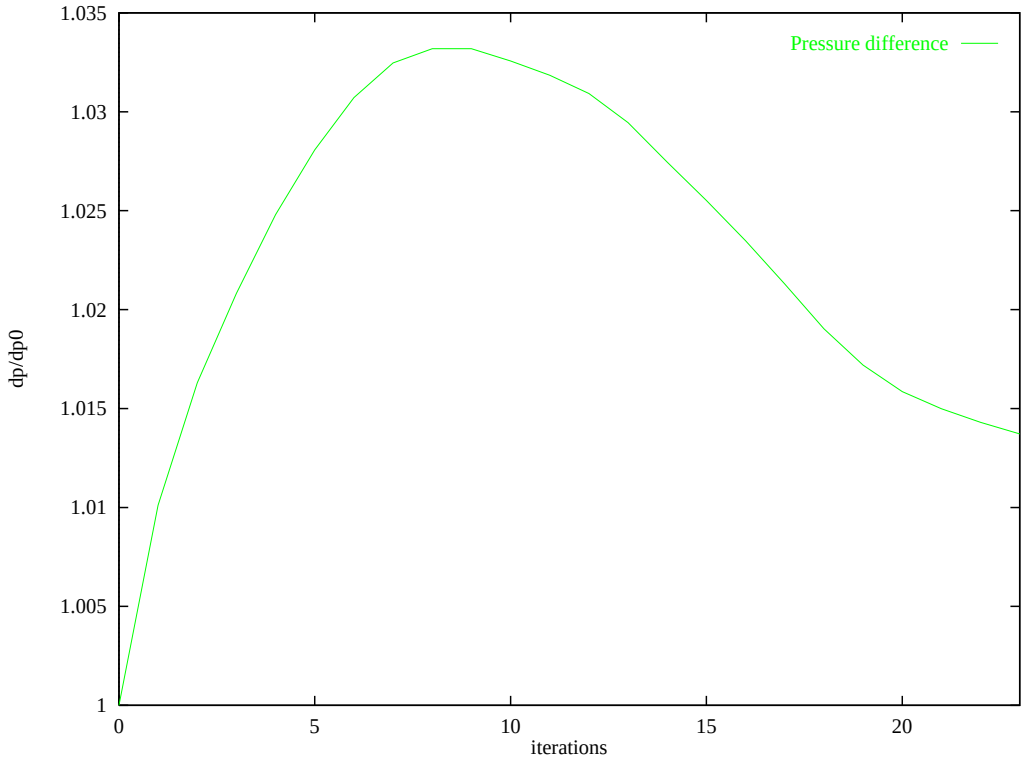

Figure 4.3: Inlet-outlet pressure difference history

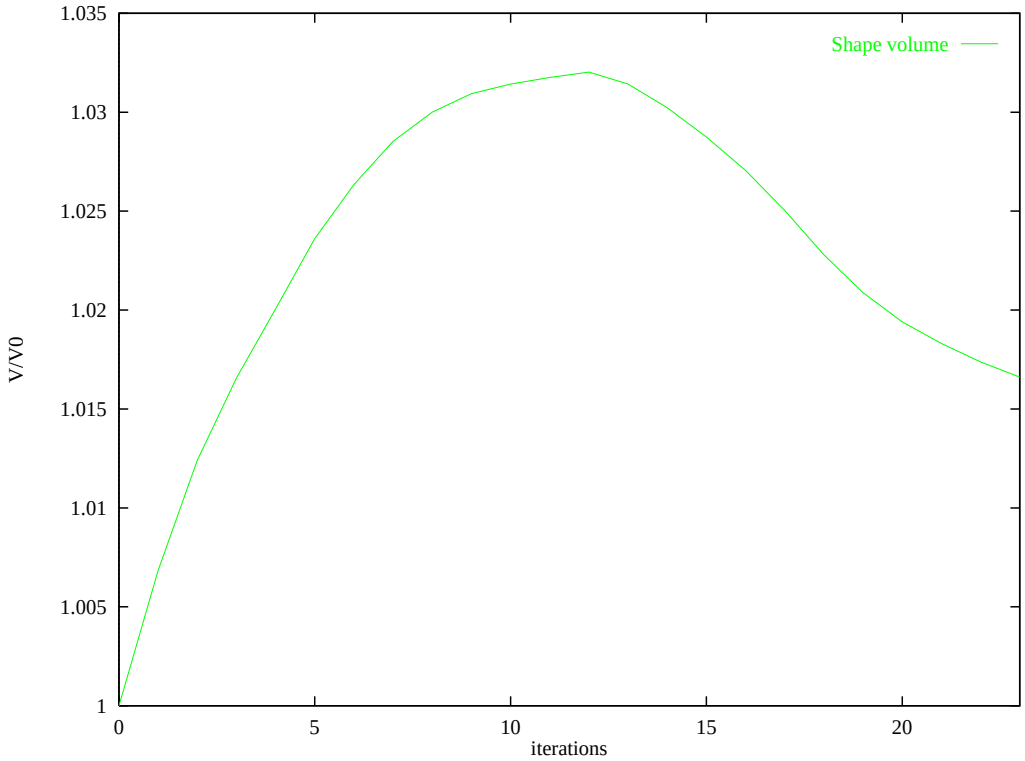

Figure 4.4: Shape volume history

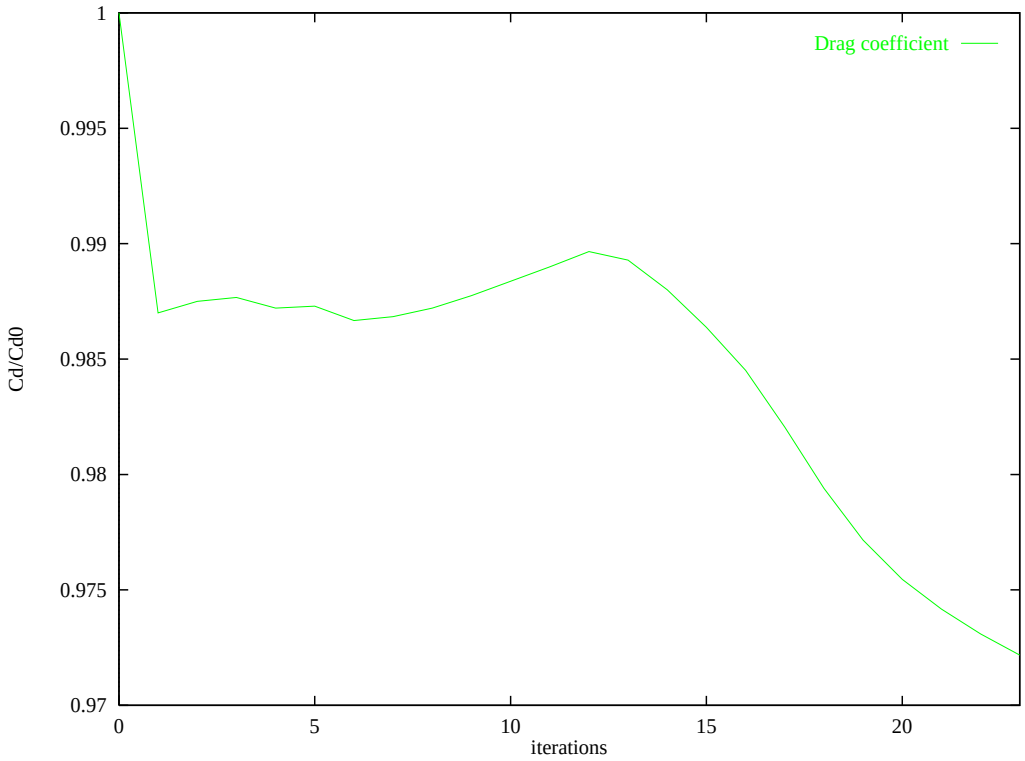

Figure 4.5: Drag coefficient history

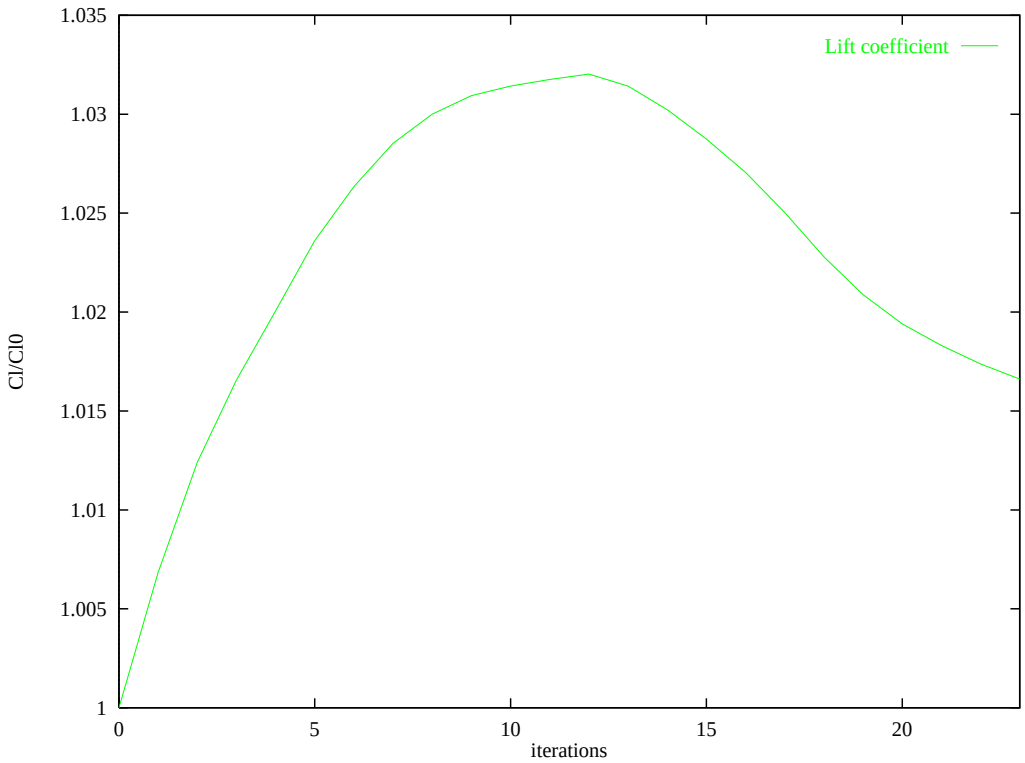

Figure 4.6: Lift coefficient history

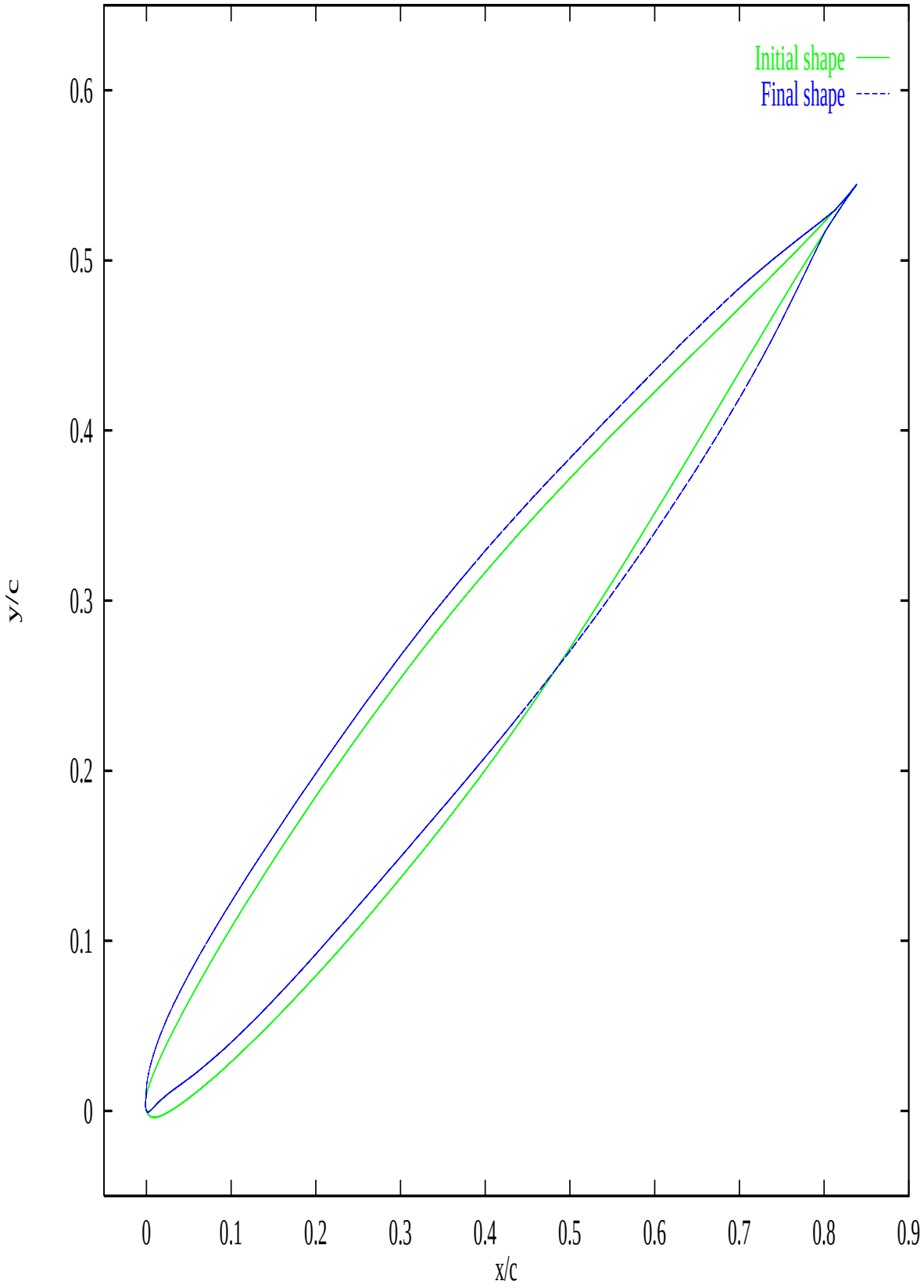

Figure 4.7: Initial and final shapes

4.2 FC1001

The previous case is a classical validation test with the NACA 65-010 profile. The profile FC-1001 is a real case from a fan with the following characteristics:

- rotational velocity $\Omega = 3500\ rpm$;
- flow rate $Q = 1400\ m^3/h$;
- number of blades $N = 11$;
- hub radius $R_h = 62\ mm$;
- shroud radius $R_s = 131\ mm$.

This case represents a medium section at a radius $R_m = 100\ mm$.

The initial data for this case are:

- initial profile FC1001;
- incompressible turbulent flow, $Re = 1.1 \cdot 10^5$, $M = 0.07$;
- stagger angle: $\beta = 75.37^o$;
- solidity: $\sigma = 1.03293$;
- number of control parameters on the shape x_c: $n_c = 174$;
- local geometrical constraints: none;
- maximal distortion allowed for the shape 0.0008;
- step size $5 \cdot 10^{-4}$;
- cost function $J = 1/C_y$;
- optimization iterations: 40;
- progressive optimizations: the intermediate solutions were considered converged when the residual decreased one order of magnitude.

The optimization results are:

- the optimization process cost was approximately 2 works;
- the cost function $J = C_d/C_l$ was reduced by 2.3 %;

- the inlet-oulet pressure difference was increased by 2.4 %;
- the shape volume was increased by 1.8 %.

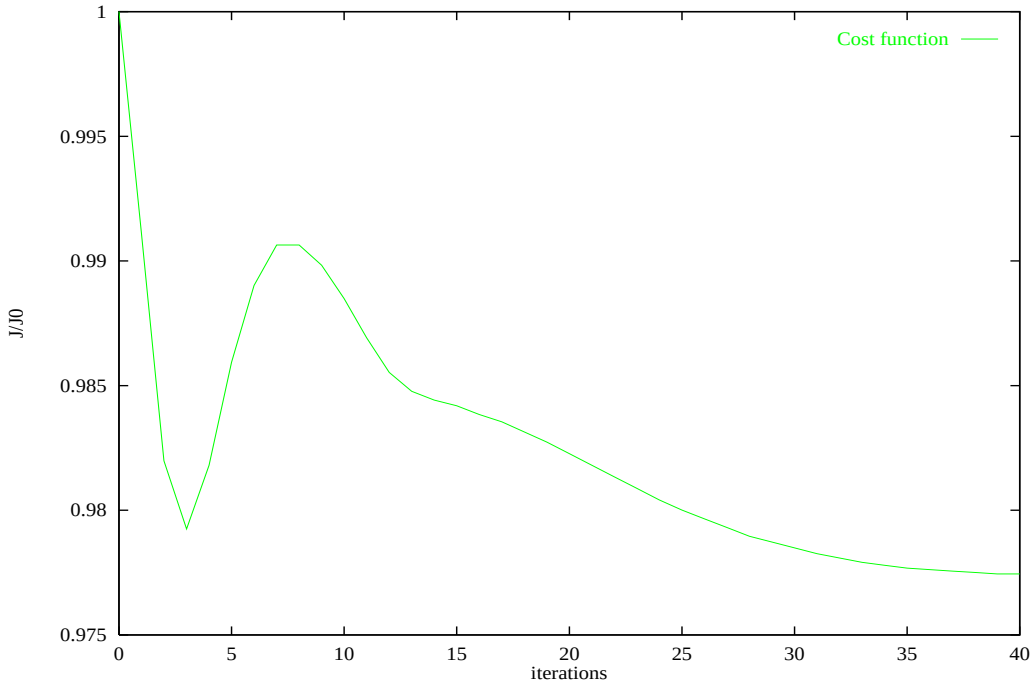

Figure 4.8: Cost function history

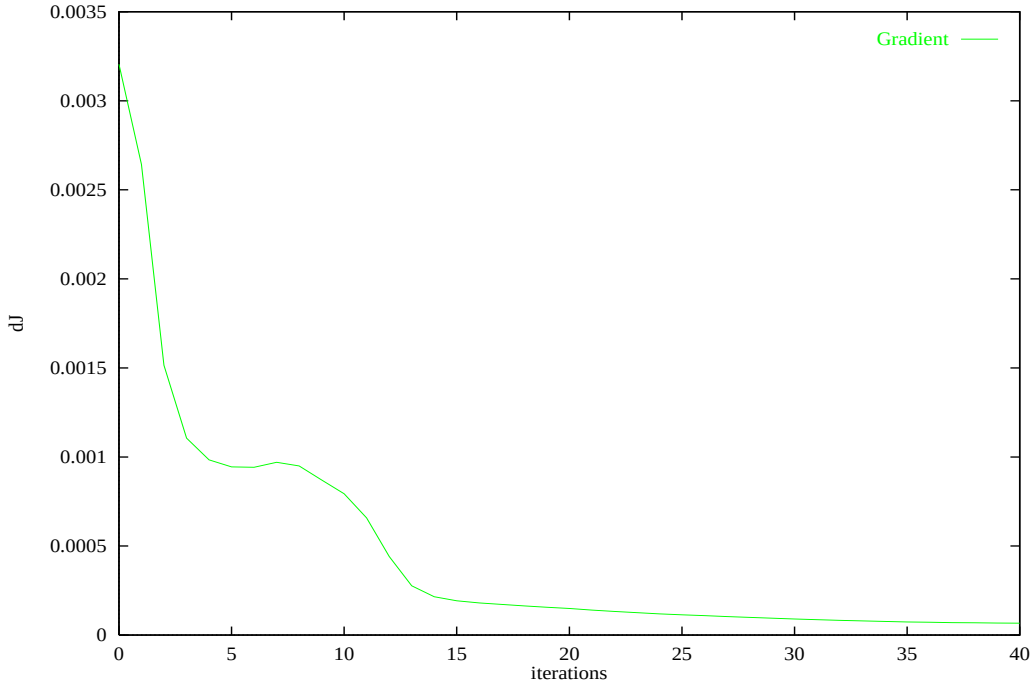

Figure 4.9: Gradient convergence

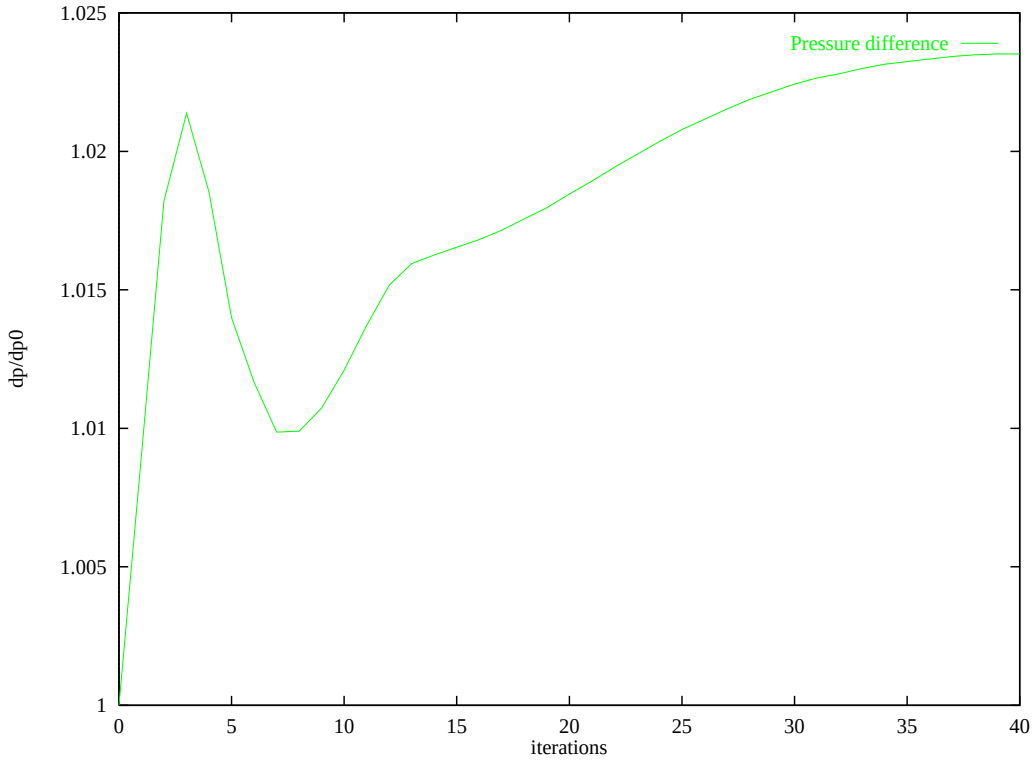

Figure 4.10: Inlet-outlet pressure difference history

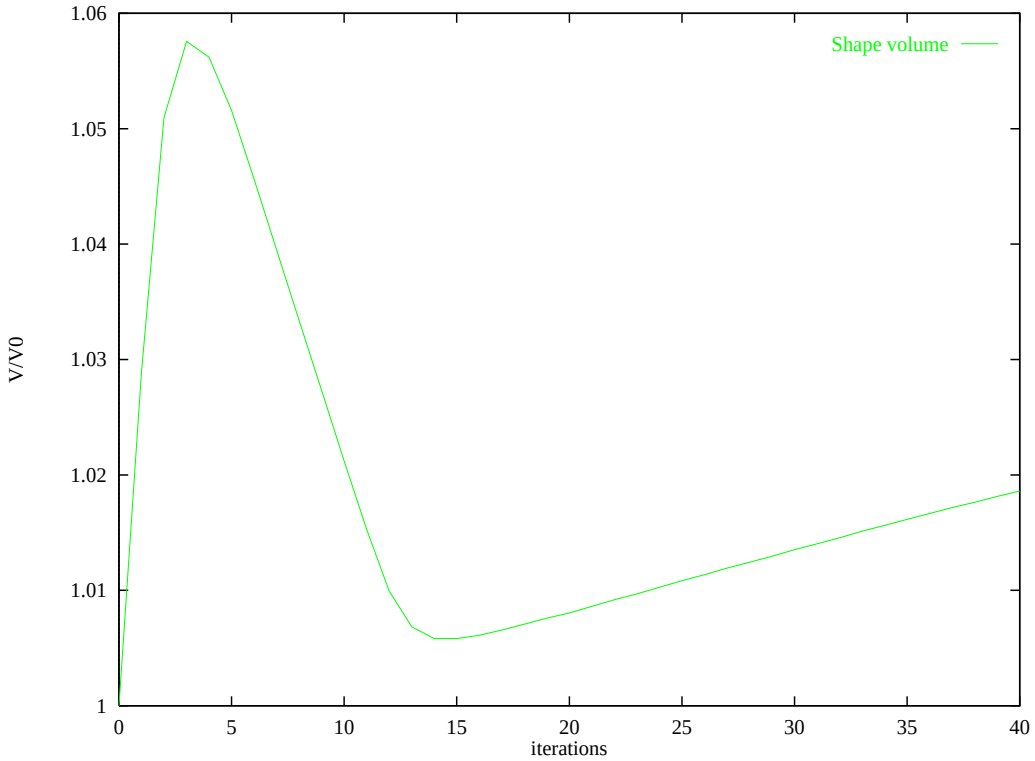

Figure 4.11: Shape volume history

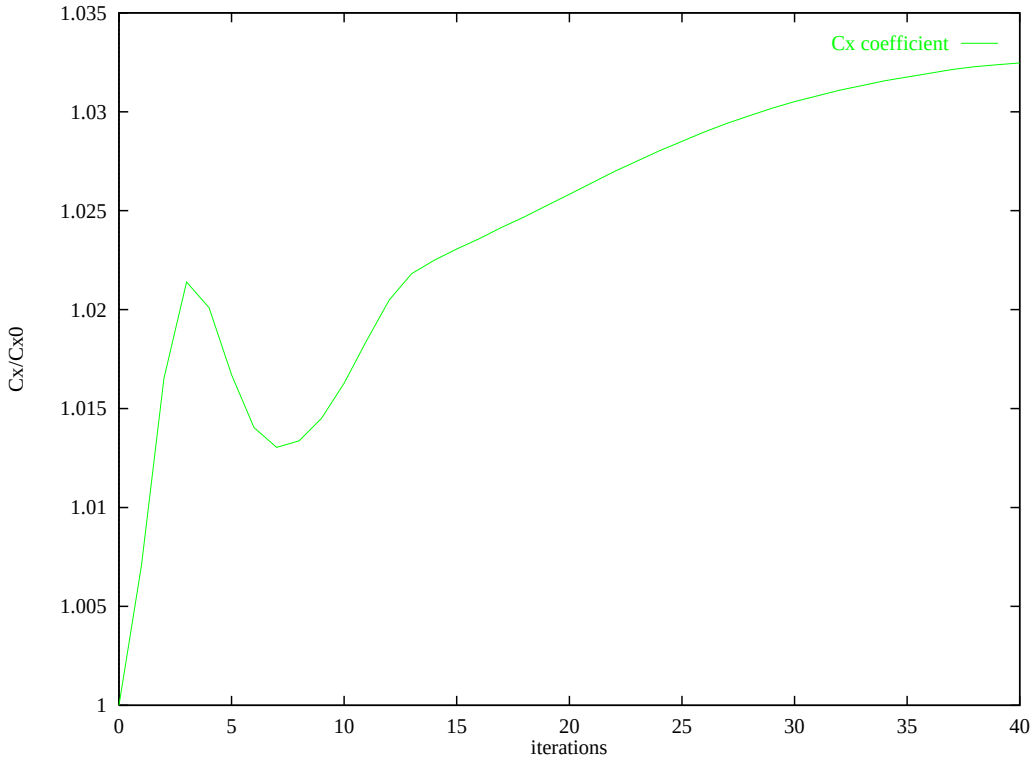

Figure 4.12: Drag coefficient history

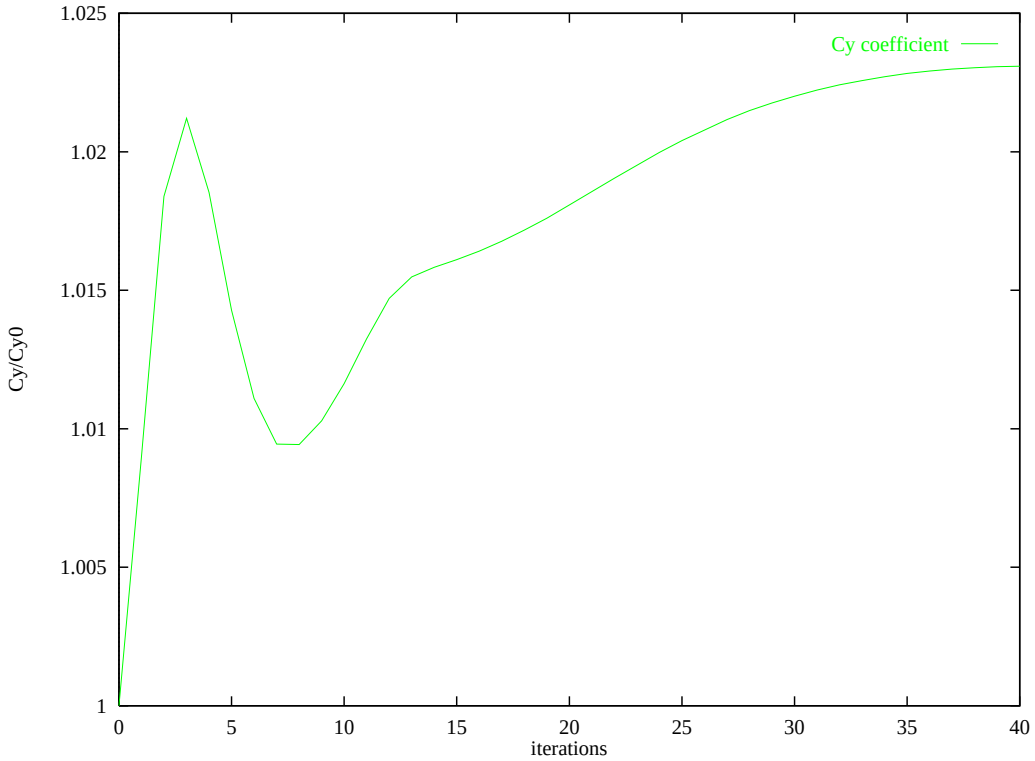

Figure 4.13: Lift coefficient history

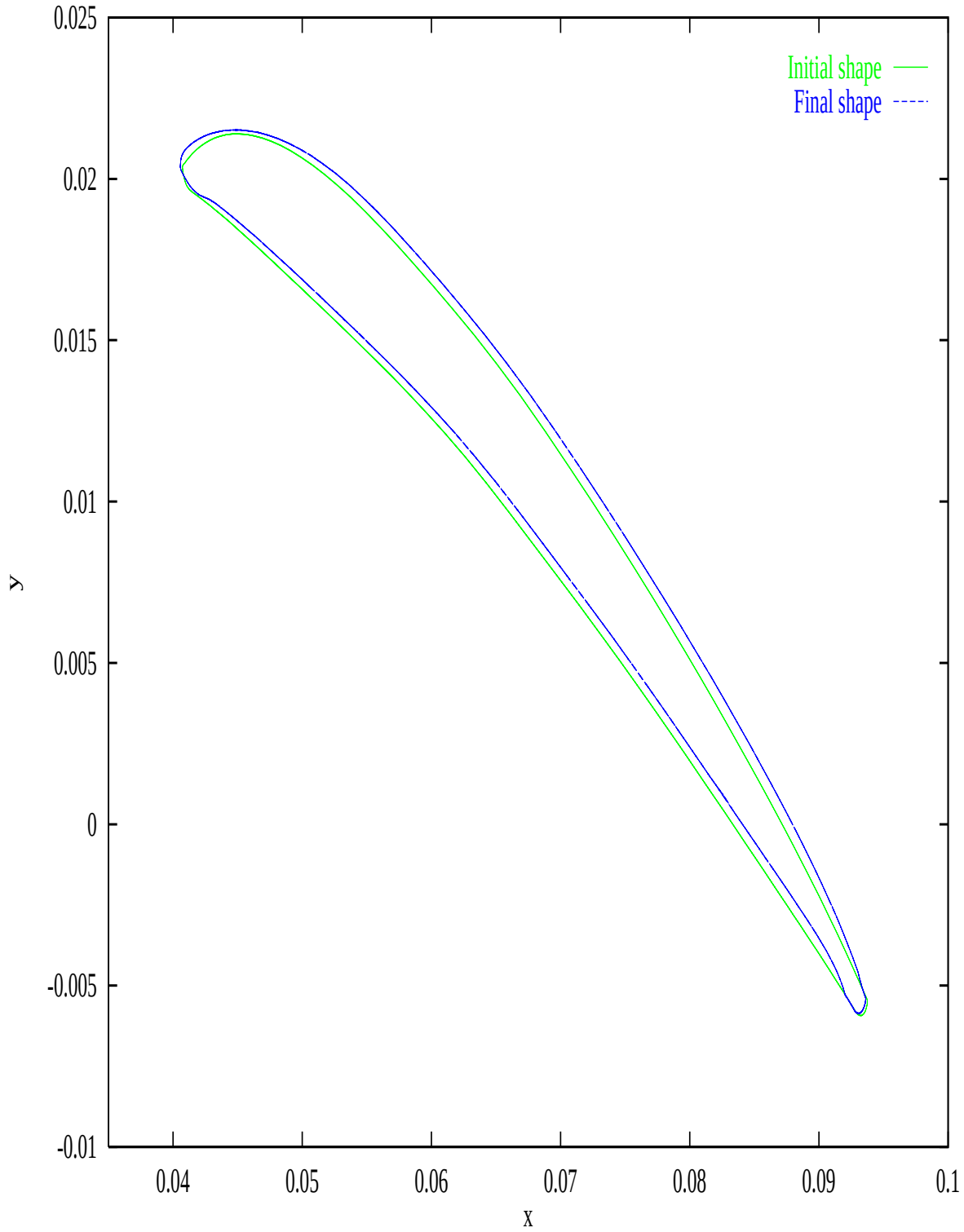

Figure 4.14: Initial and final shapes

Chapter 5

CONCLUSIONS

We developed a new code for shape optimization in turbomachinery. TASCOPT-3.0 was the results of the collaboration VALEO TM - INRIA Rocquencourt and consisted of the application of the theoretical researches on optimization obtained and validated at INRIA in an industrial environment, for the specific problems of VALEO TM.

We used a new approach for shape optimization in aerodynamics problems. The main ingredients of this approach were an unstructured CAD-free framework for shape and mesh deformations, automatic differentiation of programs for computing the discrete adjoint operator and the gradient approximation based on local informations on the shape. We extended the CAD-free framework to multi-block structured grids and coupled these techniques with the commercial flow solver TASCflow. Specific interfaces were developed between the optimization code and the flow solver. Several optimization methods based on gradient were employed and others are in study, allowing great improvements in the future code capabilities.

The optimization procedure was applied to incompressible turbulent flows around car engine cooling fan blade profiles. The presented 2D results serve as a basis for the further optimization of complex 3D configurations.

Bibliography

[1] Advanced Scientific Computing Ltd. (1995), *TASCflow User Documentation.*

[2] L.J. Herrig, J.C. Emery, J.R. Erwin (1957), *Systematic Two-Dimensional Cascade Tests of NACA 65-series Compressor Blades at Low Speeds*, NACA Technical Note 3916, Langley Aeronautical Laboratory.

[3] B.E. Launder, D.B. Spalding (1972), *Mathematical Models of Turbulence*, Academic Press.

[4] R. I. Lewis (1996), *Turbomachinery Performance Analysis*, John Wiley & Sons Inc., New York.

[5] A. Marrocco (1984), *Simulations numériques dans la fabrication des circuits à semi-conducteurs*, INRIA Research Report.

[6] G. Medic, B. Mohammadi, N. Petruzzelli, M. Stanciu and F. Hecht (1999), *3D Optimal Shape Design for Complex Flows : Application to turbomachinery*, 37th AIAA Aerospace Sciences, January 11-14, Reno, USA, AIAA-99-0833 Paper.

[7] G. Medic, B. Mohammadi, M. Stanciu and S. Moreau (1998), *A New Approach for Optimal Blade Design*, ICFD Conference on Numerical Methods for Fluid Dynamics, March 31 - April 3, Oxford, UK.

[8] G. Medic, B. Mohammadi, M. Stanciu and S. Moreau (1998), *Optimal Airfoil and Blade Design in Compressible and Incompressible Flows*, 29th AIAA Fluid Dynamics Conference, June 15-18, Albequerque, USA, AIAA-98-2898 Paper.

[9] B. Mohammadi, J.M. Malé, N. Rostaing Schmidt (1995), *Automatic Differentiation in Direct and Reverse Modes: Application to Optimum Shapes Design in Fluid Mechanics*, Proc. SIAM workshop on AD, Santa Fe, SIAM.

[10] V.C. Patel, W. Rodi, G. Scheurer (1985), *Turbulence Models for Near-Wall and Low Reynolds Number Flows: A Review*, AIAA Journal, Volume 23, Number 9.

[11] N. Petruzzelli and B. Mohammadi (1998), *Incomplete Sensitivities and BFGS method for 3D Optimal Shape Design*, INRIA Research Report.

[12] O. Pironneau (1984), *Optimal Shape Design for Elliptic Systems*, Springer-Verlag, New York.

[13] W. Rodi (1991), *Experience with Two-Layer Models combining the $k-\varepsilon$ Model with One-Equation Model near the Wall*, AIAA 91-0216, Reno, Nevada.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399