Fictitious domains with separable preconditioners versus unstructured adapted meshes

Alexandre Bespalov, Yuri Kuznetsov, Olivier Pironneau, Marie-Gabrielle Vallet

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

1992

25ème anniversaire

N° 1614

Programme 6
Calcul Scientifique, Modélisation et
Logiciel numérique par Ordinateur

FICTITIOUS DOMAINS WITH SEPARABLE PRECONDITIONERS VERSUS UNSTRUCTURED ADAPTED MESHES

Alexander BESPALOV
Yuri KUZNETSOV
Olivier PIRONNEAU
Marie- Gabrielle VALLET

Février 1992

RR_1614

FICTITIOUS DOMAINS WITH SEPARABLE PRECONDITIONERS VERSUS UNSTRUCTURED ADAPTED MESHES

Alexander Bespalov & Yuri Kuznetsov
(Academy of Sciences, Moscow, USSR)

Olivier Pironneau & Marie-Gabrielle Vallet
(Université Paris 6 & INRIA)

Abstract .

Fictitious domain approach is a technique to solve partial differential equations on arbitrary domains, Ω, with an iterative method which is preconditioned by the separable linear operator attached to the uniform grid closest to Ω. It is very fast but it uses grids which are not very smooth near the boundaries.

In order to evaluate these methods and compare them with more classical methods like the Finite Element Method on unstructured meshes and the preconditionned conjugate gradient method, we have applied two of these methods to an industrial problem for which the solution is known: the computation of the pressure distribution of an airfoil with lift in an irrotational flow.

At equal number of points the fictitious domain approach is much faster. At equal precision the fictitious domain approach is also faster; however if a mesh optimizer is used the Finite Element Method has similar execution times

DOMAINES FICTIFS AVEC PRECONDITIONNEURS SEPARABLES CONTRE MAILLAGES ADAPTES NON STRUCTURES

Résumé .

La méthode des domaines fictifs est une technique pour résoudre les équations aux dérivées partielles dans des domaines quelconques Ω en utilisant les solveurs rapides factorisés sur un préconditionneur issu du système linéaire associé au même problème mais sur un maillage uniforme voisin de celui de Ω. Ces méthodes sont très rapides mais elles utilisent des maillages assez irréguliers près des bords.

Pour évaluer cette approche et la comparer aux méthodes plus classiques qui suivent les parois comme la méthode des éléments finis, nous avons résolu par les deux techniques un problème industriel classique: le calcul de la distribution de pression autour d'un profil portant en écoulement irrotationel.

A nombre de points égal la méthode des domaines fictifs est beaucoup plus rapide. A précision égale la méthode des domaines fictifs est aussi plus rapide; toutefois avec un optimiseur de maillage les temps calcul de la méthode des éléments finis sont du même ordre de grandeur.

1. INTRODUCTION

The numerical solution of Nonlinear Partial Differential Equations on computers is expensive yet industrially important (Computational Fluid Dynamics and large displacements Structure Dynamics for examples). Numerical methods on structured meshes are usually faster than for unstructured meshes; the repetitive structures of the rows of the linear systems can be exploited and the vectorization of the programs are easier. Yet unstructured meshes are needed for industrial applications because the geometries are usually complex. Hence the success of the Finite Element Methods.

By using iterative methods to solve the linear or nonlinear systems it is possible to retain some of the advantages of the structured mesh approach on problems which involve complex geometries. One such method is to precondition the linear systems on the general mesh by the one corresponding to a structured mesh close to it. To retain the "i,j,k" structure of the linear system it is best then to construct the general mesh as follows: Inbed the domain of integration of the PDE into a larger "fictitious" domain of simple shape (a rectangle) and deform the uniform mesh by moving the nearest points onto the boundaries of the domain.

It is difficult to trace back who first proposed this method but the Russian school of applied mathematics has developped it extensively[8].Recently Young et al[17] (see also [18]) at Boeing have used the method for transonic flows in 3 dimensions and demonstrated its power in an industrial environment: easier mesh generater and faster linear solver.

The problem we want to address in this paper is the comparison between the fictitious domain approach and the more classical approach which uses smooth meshes near the boundaries. The advange of the later is that the mesh can be optimized and/or adapted with an a priori error estimator [19][20][21]. While it is obvious that the fictitious domain approach will be faster when compared at equal degree of freedom, it is not clear that it is so if the methods are compared at equal precision. Thus we have chosen an industrial problem for which we know the solution: the computation of the lift of a Joukowski airfoil. It is a linear problem but we feel that it is releveant to the nonlinear situations because these usually break the nonlinearity into a sequence of linear problems.

So for the numerical solution of this problem we have two finite element methods of degree 1:

1. The mesh is unstructured the linear system is stored in Morse form and solved by the Preconditioned Conjugate Gradient method (PCG) and the preconditioner is an Incomplete Choleski factor which keeps the elements of the neighbors of the neighbors of the vertices.
2 The mesh is uniform except for the first layer near the boundary. The linear system is also solved by a PCG but the preconditioner is the linear system corresponding to the finite difference operator on the uniform mesh including near the boundary. This operator can be factorized in x and y; hence the preconditioner is the product of two tri-diagonal linear systems.

The comparison of the methods shows the following:

- At equal number of points method 2 is much faster.

- At equal precision method 2 is also faster if the mesh of method 1 is not optimized.
- If the mesh is optimized so that the number of points can be reduced without deteriorating the precision then both methods are equally good.

In addition this paper contains a comparison of method 2 for a Neumann problem and a Dirichlet problem because the calculation of the lift of the airfoil can be done either with a potential function or a stream function. It is shown that the formulation with the Neumann condition (potential) is better.

2. THE PROBLEM.

For a perfect incompressible fluid, velocity fields u which satisfy

$$\nabla.u = 0 \tag{2.1}$$

$$\nabla \times u = 0 \tag{2.2}$$

are solutions to the general equations of fluids, the Navier-Stokes equations, if the pressure is given by

$$p = p_\infty - \frac{1}{2}|u|^2. \tag{2.3}$$

The flow around a wing profile S at rest in an unbounded fluid at constant speed at infinity u_∞ is approximated numerically by a flow in a bounded domain Ω with an outer boundary Γ_∞ at a finite distance ; so Ω is a two dimensional domain with boundary $\Gamma = \Gamma_\infty \cup S$ (see Figure 2.1).

The wake, modeled by a stream line/surface Σ issued from the trailing edge behind S, is not irrotational so (2.1)(2.2) hold in $\Omega - \Sigma$. The boundary conditions on Σ are

$$u.n_\Sigma|_\Sigma = 0 \quad \Sigma \text{ is a streamline} \tag{2.4}$$

$$(\nabla \times u).n_\Sigma|_\Sigma = 0 \quad \text{Vorticity is parallel to } \Sigma. \tag{2.5}$$

$$|u|_{\Sigma+} = |u|_{\Sigma-} \quad \text{Continuity of (2.1) and of the pressure} \tag{2.6}$$

Here n_Σ is the normal to Σ and Σ^+, Σ^- indicate the value from below and above when the functions are discontinuous.

Problem (2.1)-(2.6) is well posed with boundary conditions on the normal component of u:

$$u.n|_{\Gamma_\infty} = u_\infty.n, \quad u.n|_S = 0 \tag{2.7}$$

2.1 Stream functions.

In two dimensions, (2.1) implies that there exists a scalar function $\psi(x)$ such that

$$u = \nabla \times \psi = \{\psi_{,2}, -\psi_{,1}\}^T. \tag{2.8}$$

Then (2.2) reduces to

$$\Delta\psi = 0 \quad in \quad \Omega - \Sigma \tag{2.9}$$

In two dimensions (2.5) is automatically satisfied and (2.4), (2.6) imply the continuity of u across Σ. This, in turn, will be satisfied if (2.9) is extended into Ω and $\psi \in H^2(\Omega)$. Hence we consider the following problem:

$$\Delta\psi = 0 \quad in \quad \Omega \tag{2.10}$$

To find boundary conditions for ψ we use (2.7):

$$\frac{\partial\psi}{\partial s} = g \tag{2.11}$$

where s is the tangent direction and g is either $u_\infty.n$ or zero. This equation can be integrated and give

$$\psi|_{\Gamma_\infty} = u_{\infty 1}x_2 - u_{\infty 2}x_1 \quad \psi|_S = \beta, \text{ constant.} \tag{2.12}$$

The constant β is not known but it can be found by asking the continuity of the flow at the trailing edge P:

$$\frac{\partial\psi}{\partial n}|_{P+} = -\frac{\partial\psi}{\partial n}|_{P-} \tag{2.13}$$

It can be shown that there is one and only one solution to (2.10)(2.12)(2.13) because the solution of (2.10)(2.12) is linear in β and (2.13) is an equation to determine β. Mathematically it has been shown (see Grisvard [10] for example) that it is also the only solution of (2.10)(2.12) which belongs to $H^2(\Omega)$; when β is not such that (2.13) holds, the solution has a singularity at the trailing edge and it is in $H^1(\Omega)$ but not in $H^2(\Omega)$.

From a practical point of view the easier way to solve the problem is to use the linearity in β. Let ψ^0 and ψ^1 be the solutions of

$$\Delta\psi^0 = 0 \quad in \quad \Omega, \quad \psi^0|_{\Gamma_\infty} = u_{\infty 1}x_2 - u_{\infty 2}x_1, \quad \psi^0|_S = 0 \tag{2.14}$$

$$\Delta\psi^1 = 0 \quad in \quad \Omega, \quad \psi^1|_{\Gamma_\infty} = u_{\infty 1}x_2 - u_{\infty 2}x_1, \quad \psi^1|_S = 1. \tag{2.15}$$

Then the solution to the problem is

$$\psi = \beta\psi^1 + (1-\beta)\psi^0 \tag{2.16}$$

with β such that (2.13) holds, that is

$$\beta = -\frac{\frac{\partial \psi^0}{\partial n}|_{P+} + \frac{\partial \psi^0}{\partial n}|_{P-}}{\frac{\partial(\psi^1-\psi^0)}{\partial n}|_{P+} + \frac{\partial(\psi^1-\psi^0)}{\partial n}|_{P-}} \tag{2.17}$$

2.2 Potential flow.

Another approach is to say that (2.2) implies that there exists φ such that

$$u = \nabla\varphi \tag{2.18}$$

Then (2.1) implies that φ satisfy

$$\Delta\varphi = 0 \ \ in \ \ \Omega - \Sigma. \tag{2.19}$$

Boundary conditions are

$$\frac{\partial\varphi}{\partial n} = u_\infty.n \quad on\ \Gamma_\infty \quad \frac{\partial\varphi}{\partial n} = 0\ on\ S \tag{2.20}$$

Unfortunately, φ cannot be extented continuously across Σ and (2.4)-(2.6) be satisfied. In two dimensions there is a jump discontinuity of φ across Σ but the jump β is constant. Then it is easy to see that (2.6) needs be written only at the trailing edge P. So we obtain the following condition on Σ :

$$\varphi|_{\Sigma+} = \varphi|_{\Sigma-} + \beta \tag{2.21}$$

where the constant β is determined by writing that

$$\frac{\partial\varphi}{\partial s}|_{P+} = \frac{\partial\varphi}{\partial s}|_{P-}. \tag{2.22}$$

Here again, in practice, the linear dependency of φ upon β can be used and one computes the solutions of (2.19)(2.20)(2.21) for $\beta = 0$ and $\beta = 1$ and find β by writing (2.22).

This approach can be generalized to three dimensions.

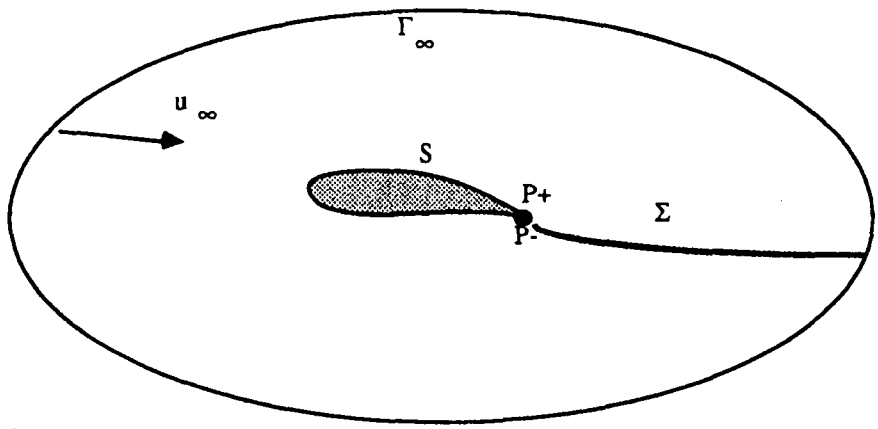


Figure 2.1 *Flow around an airfoil;* Σ *is the wake.*

For both approaches it can be shown that the lift C_f of the airfoil (the vertical component of the resultant of the force applied by the fluid on S while neglecting the viscous effects) is proportional to β :

$$C_f = \beta\rho|u_\infty| \tag{2.23}$$

where ρ is the density of the fluid.

2.3 Joukowski airfoils

An analytical solution to the above problem exists for certain profiles S.
Let

$$\xi = \xi_1 + \iota\xi_2; \quad z(\xi) = x_1 + ix_2 = \xi + \frac{C^2}{\xi.} \tag{2.24}$$

Let $\epsilon < 1$. When $\xi = C(1+\epsilon)e^{i\theta} - \epsilon C$ and $\theta \in [0, 2\pi]$, z defines a curve which is a symmetric "Joukowski" airfoil. It is a conformal transformation so the problem can be solved in ξ instead of x.

The velocity on S is (Krisnamurthy [3])

$$|u|_{|S} = \frac{2|u_\infty|}{\tau(\theta)}[\sin\alpha - sin(\alpha - \theta)] \tag{2.25}$$

where α is the angle of attack and

$$\tau(\theta) = [(\frac{C^2 + a^2 - C^2 b}{b^2 + a^2})^2 + (\frac{C^2 a}{b^2 + a^2})^2]^{\frac{1}{2}} \tag{2.26}$$

with

$$a = 2C^2(1+\epsilon)^2 \sin\theta\cos\theta - 2\epsilon(1+\epsilon)C^2\sin\theta \tag{2.27}$$

$$b = C^2(1+\epsilon)^2(\cos^2\theta - \sin^2\theta) + \epsilon^2 C^2 - 2\epsilon(1+\epsilon)C^2\cos\theta \tag{2.28}$$

In our numerical experiments we have chosen

$$\epsilon = 0.1 \quad C = \frac{30}{121} \tag{2.29}$$

The relative error between an approximate solution u_h and the exact solution is measured by

$$precision = \frac{[\int_\Gamma |u_h.s^2 - u.s^2|^2 d\gamma]^{\frac{1}{2}}}{[\int_\Gamma |u.s|^4 d\gamma]^{\frac{1}{2}}} \tag{2.30}$$

where s is the tangent vector to S.

3. SOLUTION BY THE FINITE ELEMENT METHOD ON UNSTRUCTURED MESHES

3.1 Variational formulation and approximation

Consider the formulation of the problem with a stream function:

$$\Delta\psi = 0 \quad in \quad \Omega \tag{3.1}$$

with Dirichlet boundary condition given by (2.12)

$$\psi|_\Gamma = \psi_\Gamma \tag{3.2}$$

The variational formulation of the problem is in the Sobolev space of square integrable functions with square integrable derivatives $H^1(\Omega)$. We denote as usual by $H_0^1(\Omega)$ the subset of such functions which are equal to zero at the boundary:

$$\int_\Omega \nabla\psi\nabla w = 0 \quad \forall w \in H_0^1(\Omega), \quad \psi - \psi_\Gamma \in H_0^1(\Omega). \tag{3.3}$$

We approximate the problem by

Find ψ_h such that $\psi_h - \psi_{\Gamma h} \in H_{0h}$ and

$$\int_\Omega \nabla\psi_h\nabla w_h = 0 \quad \forall w_h \in H_{0h} \tag{3.4}$$

where H_h is the conforming finite element space of degree one. Ω is divided into triangles $\{T_k\}_{1\ldots K}$ such that

- $T_k \cap T_l = \oslash$, or 1 vertex, or 1 whole side (resp. side or face) when $k \neq l$
- The vertices of the boundary of $\cup T_k$ are on Γ
- The singular points of Γ (corners) are on the boundary Γ_h of $\cup T_k$.

We note that $\Omega_h = \cup T_k$, $\Gamma_h = \partial \cup T_k$, $\{q^i\}_1^{ns}$ are the vertices of the triangles, and h is the longest side of a triangle :

$$h = \max_{q^i, q^j \in T_k} |q^i - q^j| \tag{3.5}$$

$$H_h = \{w_h \in C^0(\Omega_h) : w_h|_{T_h} \in P^1\} \tag{3.6}$$

where P^1 denotes the space of polynomials of degree 1 and C^0 the space of continuous functions. We denote by N the number of interior vertices.

3.2 Numerical solution

Problem (3.4) is a $N \times N$ positive definite linear system with respect to the values of φ_h on the vertices of the triangulation:

$$A\Phi = f \tag{3.7}$$

We have solved it by a preconditionned conjugate gradient algorithm

Algorithm 1

0 . Initialization:

Choose $C \in R^{N\times N}$ positive definite (preconditioning matrix), ϵ small positive, Φ^0 and set $g^0 = h^0 = -C^{-1}(A\Phi^0 - b)$, $n = 0$.

1 Calculate :

$$\rho^n = \frac{< g^n, h^n >_C}{h^{nT} A h^n} \tag{3.8}$$

$$\Phi^{n+1} = \Phi^n + \rho^n h^n \tag{3.9}$$

$$g^{n+1} = g^n - \rho^n C^{-1} A h^n \tag{3.10}$$

$$\gamma^n = \frac{||g^{n+1}||_C^2}{||g^n||_C^2} \tag{3.11}$$

$$h^{n+1} = g^{n+1} + \gamma^n h^n \tag{3.12}$$

2 If $|| g^{n+1} ||_C < \epsilon$, stop else increment n and go to 1.

The choice of C is critical for speed. We have tried for C the diagonal of A and the incomplete Choleski factorization (Maejerinck-Van der Worst [11]). We have also tried the incomplete Choleski factorization of A where one keeps all the elements A_{ij} which correspond to two vertices q^i, q^j which are neighbor to a third vertex q^k i.e. the non-zero entries of the incomplete factorization are the set of indices $\{i, j\}$ for which there exists k such that q^i, q^k and q^k, q^j are edges of the triangulation. This choice is the best if this preconditionner is further processed so that only the large elements are kept. Thus the precise definition of C is via the Choleski factorization algorithm as follows:

0. Choose a cut-off number $r \in]0, 1[$;

1 . *Loop i*

$$L'_{ii} = [A_{ii} - \sum_{k<i} L'_{ik}]^{\frac{1}{2}} \tag{3.13}$$

Loop $j < i$:

$$if A_{ij}^2 \neq 0 \ then \ \ L'_{ij} = \frac{1}{A_{jj}}[A_{ij} - \sum_{k<j} L'_{ik} L'_{jk}] \tag{3.14}$$

End loops

2 . *Loop i,j,* $j \leq i$:

if $L'_{ij} < r \min(L'_{ii}, L'_{jj})$ *then* $L_{ij} = 0$ *else* $L_{ij} = L'_{ij}$

End loop

3 . Define $C = LL^T$.

The test on A_{ij}^2 is due to the fact that $A_{ij}^2 \neq 0$ is almost equivalent to the existence of k such that q^i, q^k and q^k, q^j are edges of the triangulation.

3.3 Practical implementation.

The Dirichlet condition is implemented via penalty. The matrix A is constructed for all indices corresponding to a vertex. Then if q^i is on the boundary A_{ii} is replaced by $10^{30}A_{ii}$. Similarly the i^{th} element on the right hand side is set to $10^{30}\psi_{\Gamma i}$

The matrices are stored in Morse form (only the non-zero elements are stored).

The conjugate gradient algorithm is stopped when the relative residual error is less than 10^{-6}.

For the Joukowski condition, experience shows that one can apply condition (2.13) by replacing P^+ and P^- by the triangles which are on S and have P as a vertex.

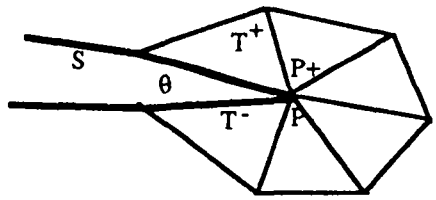


Figure 3.1 *Configuration of the triangles near the trailing edge.*

3.4 Mesh Adaption

In order to accelerate the iterative process, adaptive meshes are used to decrease the matrix size without loss of accuracy. A first computation is performed on a preliminar coarse grid, then an a posteriori error is analysed. This error defines a Riemanian metric on the computational domain. Finally, a new mesh is generated by an algorithm of Voronoi type, applied into this metric.

We first present the mesh generation process which, once the metric is given, is independant of the idea of adaption and will generate triangles as equilateral (in the given metric) as possible and of a given area.

The Delaunay-Voronoi type mesh generation process is based on a point insertion process which connects vertices to obtain triangular elements. The Delaunay criterion tends to produce triangles as equiangular as possible; it prescribes that no vertex of a triangle may be contained in the circumcircle from any other triangle; this can be proved [12] to be equivalent to the fact that this triangulation maximises the minimum of the six angles in any pair of adjacent triangles. Lawson's algorithm [13] based on diagonal swapping is applied in order to satisfy this property; for each pair of triangles which constitute a convex quadrilateral, we examine the two positions for the diagonal and choose the one satisfying the Delaunay property.

The first set of nodes consists of boundary nodes generated in such a way that the length of boundary edges is (in the metric) a given length. Then a first mesh is being generated connecting these nodes and new points are added at the center of triangles as soon as their area is larger than a given area.

When applied in the Euclidean metric, the previous algorithm produces triangular elements as equiangular as possible for a set of vertices with constant density. Adaptation is

introduced by a change of metric. Depending on it, points density and elements stretching vary on the domain.

Let $M = M(P)$ be a $2 \times 2-$matrix, positive definite, continuous for all point P. It determines on the plan a Riemanian metric. The length of a parametric curve $\{\gamma(\tau), \tau \in [0,1]\}$ in the new metric is the value of

$$\int_0^1 \sqrt{\gamma'(\tau)^\top M(\gamma(\tau))\,\gamma'(\tau)}\,d\tau.$$

Assuming that the variation of M is smooth on the domain, we can locally considere $M(P)$ as a constant. Then the norm $\|.\|_M$ associated with the new metric is (locally) defined by

$$\|X\|_M^2 = X^\top M X,\ \forall X \in \Omega.$$

The matrix M being symmetric, its eigenvalues are real and its eigenvectors are orthonormal. If we denote by λ_1, λ_2 (resp. d_1, d_2) the eigenvalues (resp. the eigenvectors) of M, then a unit vector for the norm $\|.\|_M$ will be measured by $1/\sqrt{\lambda_k}$ for the euclidian norm, if it is in the direction $d_k, (k = 1, 2)$.

During mesh generation, all measures are understood in the sense of this metric, i.e. :

- the length of boundary edges which controls boundary discretization,
- the elements area controlling the density of nodes,
- the quality criterion (Delaunay criterion) controlling the elements shape.

We now discuss how the matrix M is defined in order to satisfy an adaption criterion. The mesh adaption is based on an error estimate for a selected key variable σ of the problem; more precisely, for all examples considered here the metric is determined in order to equidistribute the interpolation error, or the maximum of its gradient. We assume that a first solution of the problem under consideration has been computed on a given mesh (background mesh) and we assume also that continuous piecewise linear approximations of the different variables are applied, so, for the key variable σ, the interpolation error depends on second derivatives of σ.

We first consider the interpolation error on an edge a_i of the triangulation

$$E_i = \left(\frac{1}{h_i}\int_{a_i} (\sigma - \hat{\sigma})^2\,dx\right)^{1/2}$$

where $\hat{\sigma}$ is the linear interpolant of σ and h_i denotes the length of a_i. It can be shown [14] that

$$E_i \simeq h_i^2 \left|\sigma''|_{a_i}\right|$$

with $\sigma''|_{a_i}$ an approximation of the second derivative of σ in the direction of a_i. We introduce the Hessian matrix of σ; it can be splitted according to its eigenvectors:

$$H = \begin{pmatrix} \partial^2\sigma/\partial x^2 & \partial^2\sigma/\partial x\partial y \\ \partial^2\sigma/\partial y\partial x & \partial^2\sigma/\partial y^2 \end{pmatrix} = R \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} R^\top.$$

And matrix M is defined by

$$M = R \begin{pmatrix} |\lambda_1| & 0 \\ 0 & |\lambda_2| \end{pmatrix} R^\top.$$

Then we have

$$E_i \simeq |a_i^\top H a_i| \leq a_i^\top M a_i.$$

We consider that the mesh will be "optimal" if the error is equidistributed; so we will choose the length h_i of each edge satisfying the condition :

$$a_i^\top M a_i = c_o.$$

that is, the mesh is equilateral and of edge length $\sqrt{c_0}$, in the metric defined by M.

Another error estimate can be equidistributed, which gives better results for numerical tests further performed. It is the elementary gradient error

$$E_K = \max_{x \in K} \|\nabla(\sigma - \hat{\sigma})(x)\|$$

It is proved in [15] that, if $\lambda_1 \lambda_2 > 0$, an optimal mesh is obtained by the construction of an equilateral mesh, for the metric associated with

$$M = H^\top H.$$

This error estimator is more sensitive to variations of σ.

Mesh adaption is based on the interpolation error of a key variable σ. The obtained mesh depends heavily on the choice of this variable. From our experience in adaption for CFD equations, we know that the velocity (or Mach number) gives good results. So we have taken $\sigma = \|\nabla \times \psi\|$, although it is not approximated by a continuous piecewise linear function in the discretised problem.

For the considered computational cases, the error E_i does not varies enough to produce meshes which are fine near the body and coarse far from it. That's why adapted meshes are generated with the metric $M = H^\top H$, that is, the gradient error is equidistributed.

Finally, we have to take care that the trailing edge is not a singularity for ψ, but it is one for both intermediate solutions ψ^0 and ψ^1. So, an adapted mesh for the key variable $\sigma = \|\nabla \times \psi\|$ will not be refined near the trailing edge of the profile and solutions ψ^0 and ψ^1 will not be accurate there. Consequently, computation of the coefficient β can't be precise. To avoid that, adaption has been performed for the solution ψ^1 of problem (2.15). Thus, obtained meshes are coarse except around trailing and leading edges.

The initial mesh was a previous coarse mesh around a NACA airfoil. An adapted mesh has been created for the nonlifting case, but around the Joukowski airfoil. A third mesh has been adapted for that case, and resolutions have been performed for lifting cases. Then the last adapted mesh has been generated for each computational case.

3.4 Results .

In all tests the precision on the residual is set to 10^{-6} in the conjugate gradient algorithm. The cut-off number r is set to 0.1. The test have been run from Fortran on an Apollo workstation DN 4000.

3.4.1 Angle of attack=0.

First a mesh is built around the Joukowski airfoil described in Section 2.3 at zero angle of attack with the mesh generator EMC^2 (Hecht-Saltel [16]) (Figure 3.2) and a computation is done to obtain a criterium for the mesh adaption. The following performances are found

- Precision 3.15%
- Number of vertices: 2503
- Number of triangles: 4888
- Number of iterations of conjugate gradients: 34
- Computing time 80" of which 26" is for building C

Thus this would be the computing time if no mesh optimization where performed. Then the mesh of Figure 3.3 is obtained with mesh adaption. The problem is solved again and the following performances are found (Figure 3.4):

- Precision 3.12%
- Number of vertices: 1063
- Number of triangles: 2010
- Number of iterations of conjugate gradients: 23
- Computing time 23" of which 8" is for building C

(Notice that only one Dirichlet problem needs to be solved in the nonlifting case because $\beta = 0$).

3.4.2 Angle of attack= 2°

The initial mesh of 3.4.1 is also used to find a better mesh with the mesh adaption program (Figure 3.5). Then the following performances are obtained (Figure 3.6):

- Precision 2.3%
- Number of vertices: 1584
- Number of triangles: 3076
- Number of iterations of conjugate gradients: 21 and 18
- Computing time 40" of which 11" is for building C

3.4.2 Angle of attack= 5°

The initial mesh of 3.4.1 is also used to find a better mesh with the mesh adaption program. Then the following performances are found (Figure 3.7):

- Precision 3.9%
- Number of vertices: 1545

- Number of triangles: 3001
- Number of iterations of conjugate gradients: 18 and 20
- Computing time 38" of which 10" is for building C

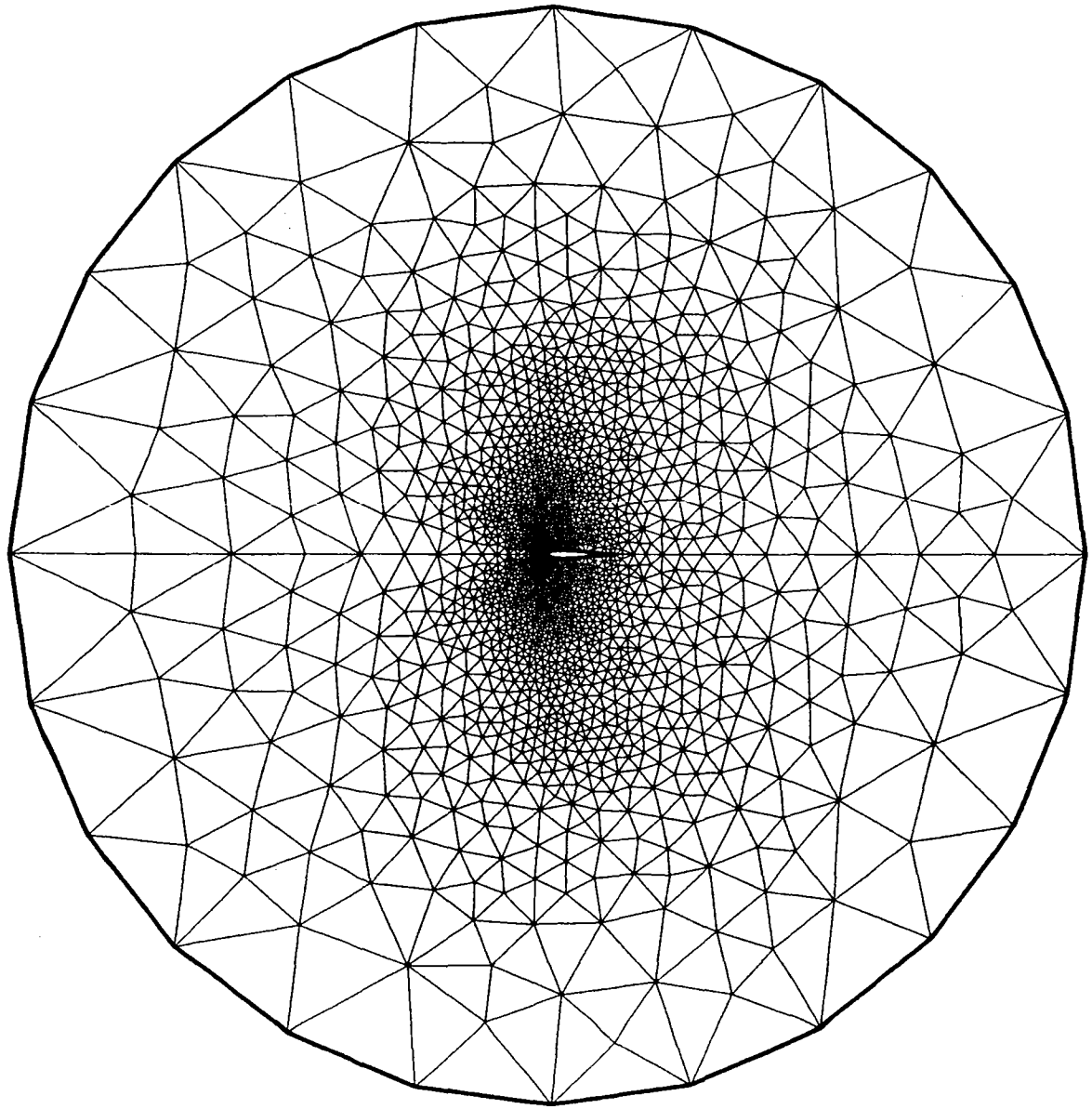


Fig 3.2a: Initial mesh before adaption: 2503 vertices and 4888 triangles.

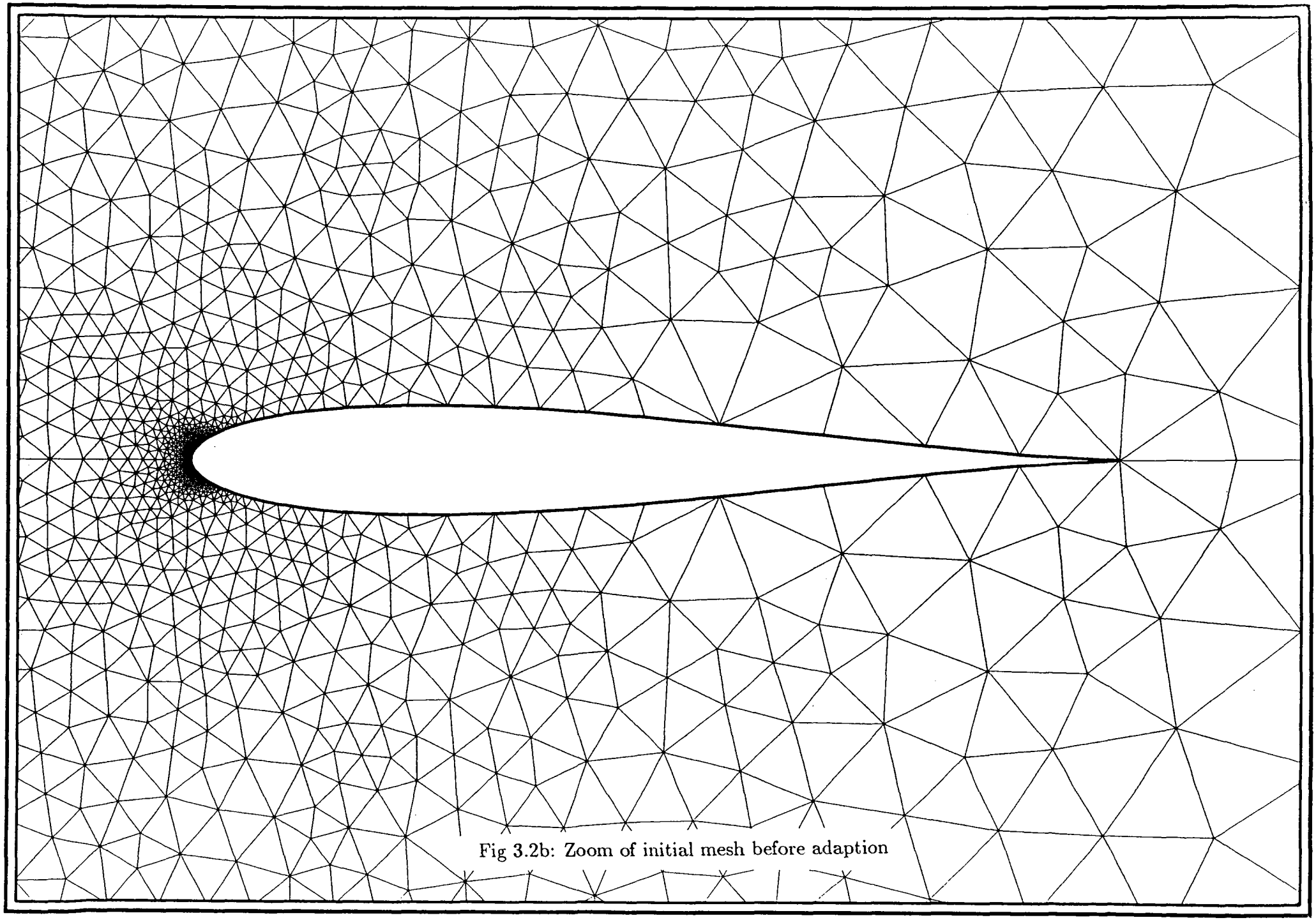
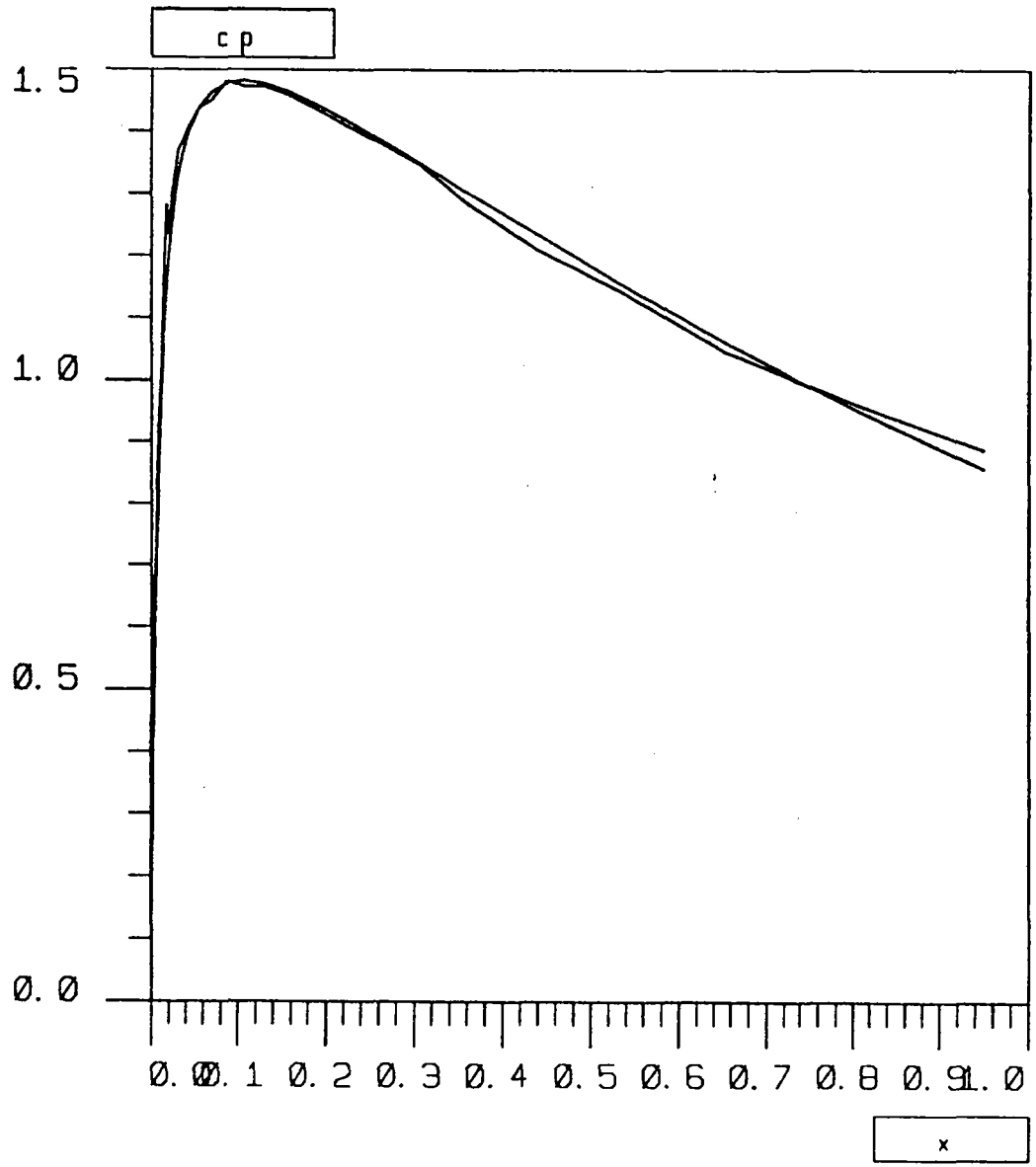


Fig 3.2b: Zoom of initial mesh before adaption

Fig 3.3: Final mesh after adaption: 1063 vertices and 2010 triangles.

Fig 3.4: Pressure coefficient on the airfoil versus x for the nonlifting case.



MODULEF : mvallet

alpha=0. ns=1063 tps=23 err=3.12%

19/11/91

cp0.dta

NOMBRE DE COURBES : 2

EXTREMA EN X :

0.58E-16 0.95

EXTREMA EN Y :

0.64E-30 1.5

TRACE DE COURBES

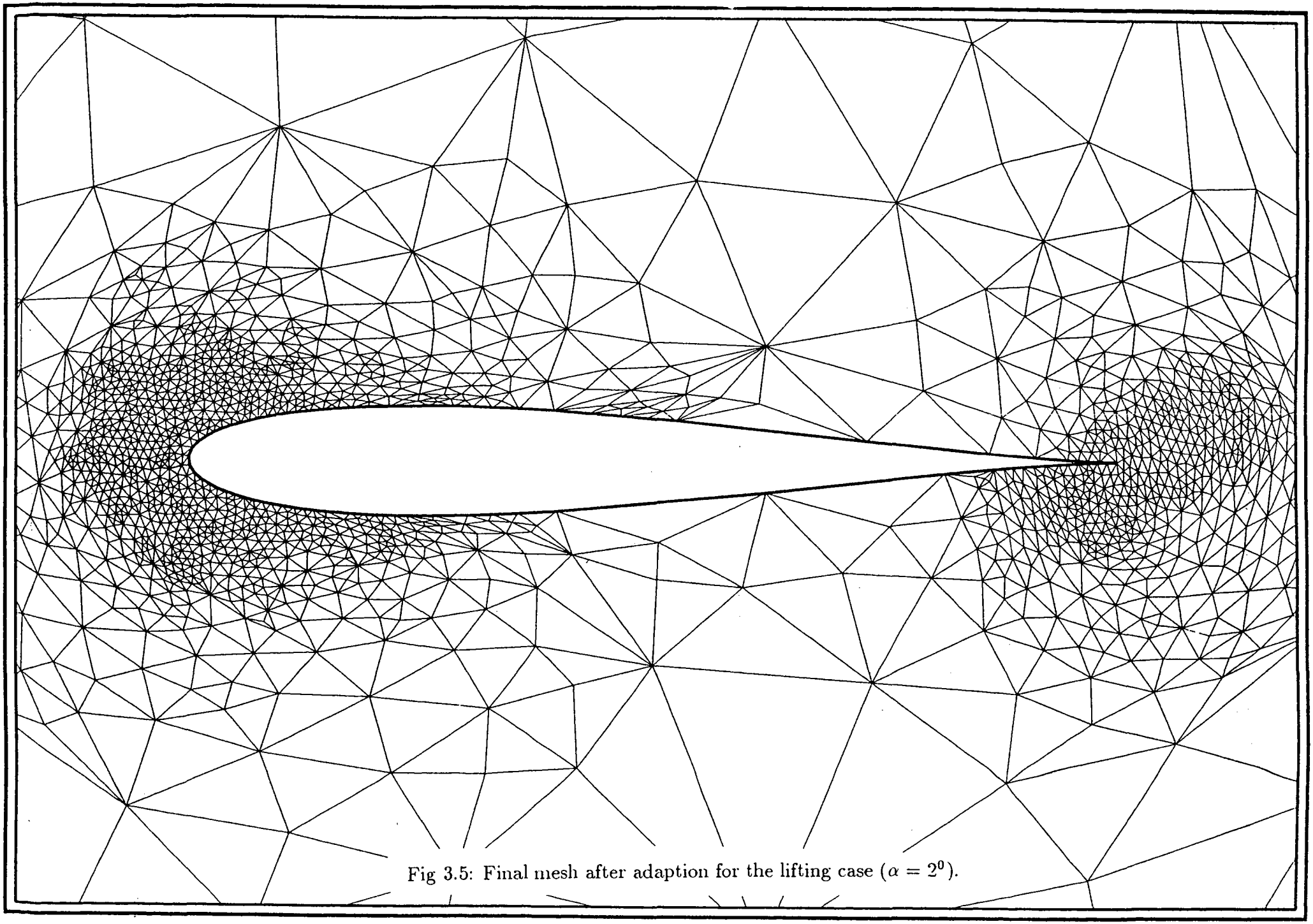
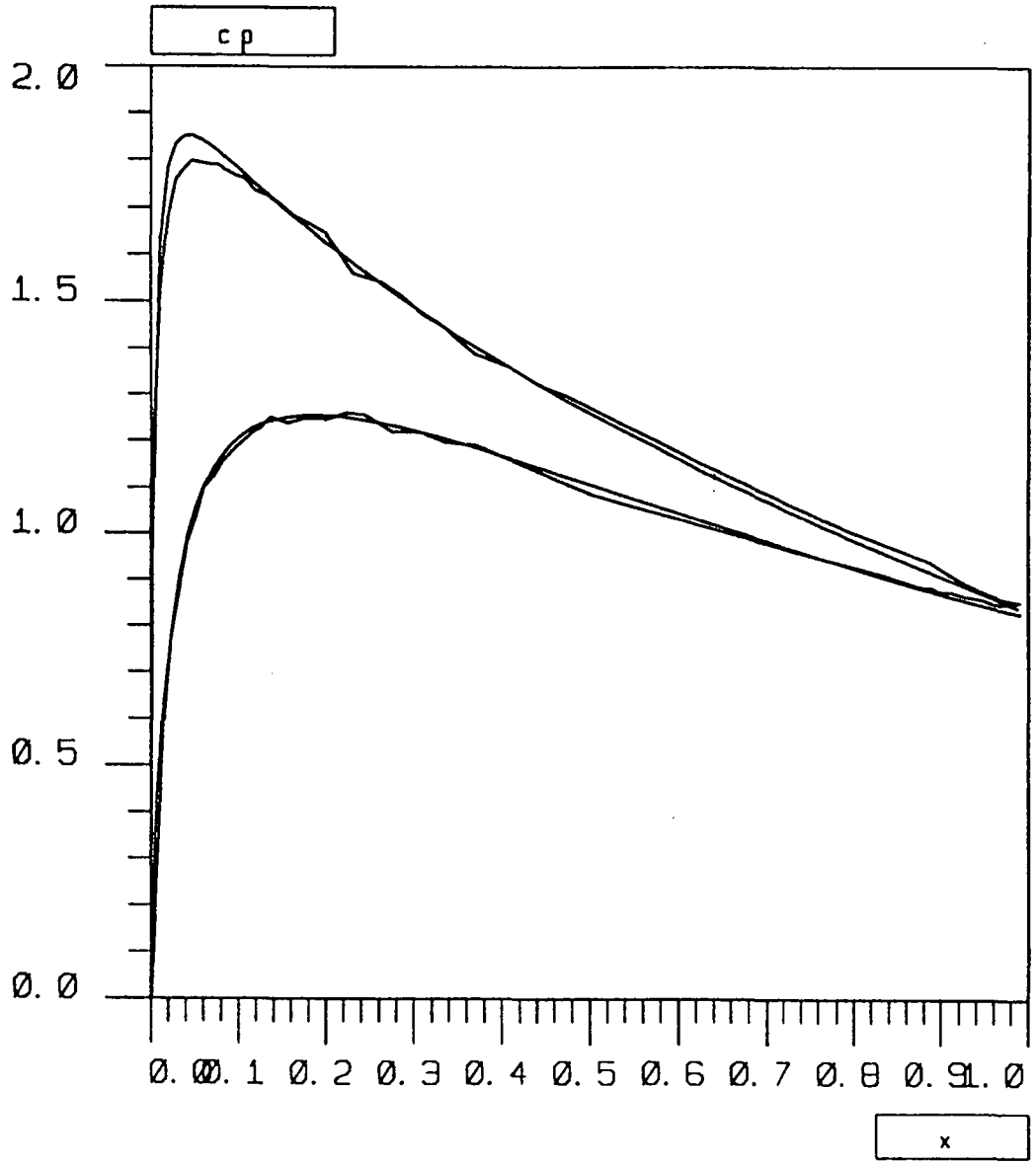


Fig 3.5: Final mesh after adaption for the lifting case ($\alpha = 2^0$).

Fig 3.6: Pressure coefficient on the airfoil versus x for the lifting case ($\alpha = 2^0$).



MODULEF : mvallet

alpha=2. ns=1584 tps=53 err=2.12%

19/11/91

keepp/jouk321.2.cp0

NOMBRE DE COURBES : 2

EXTREMA EN X :

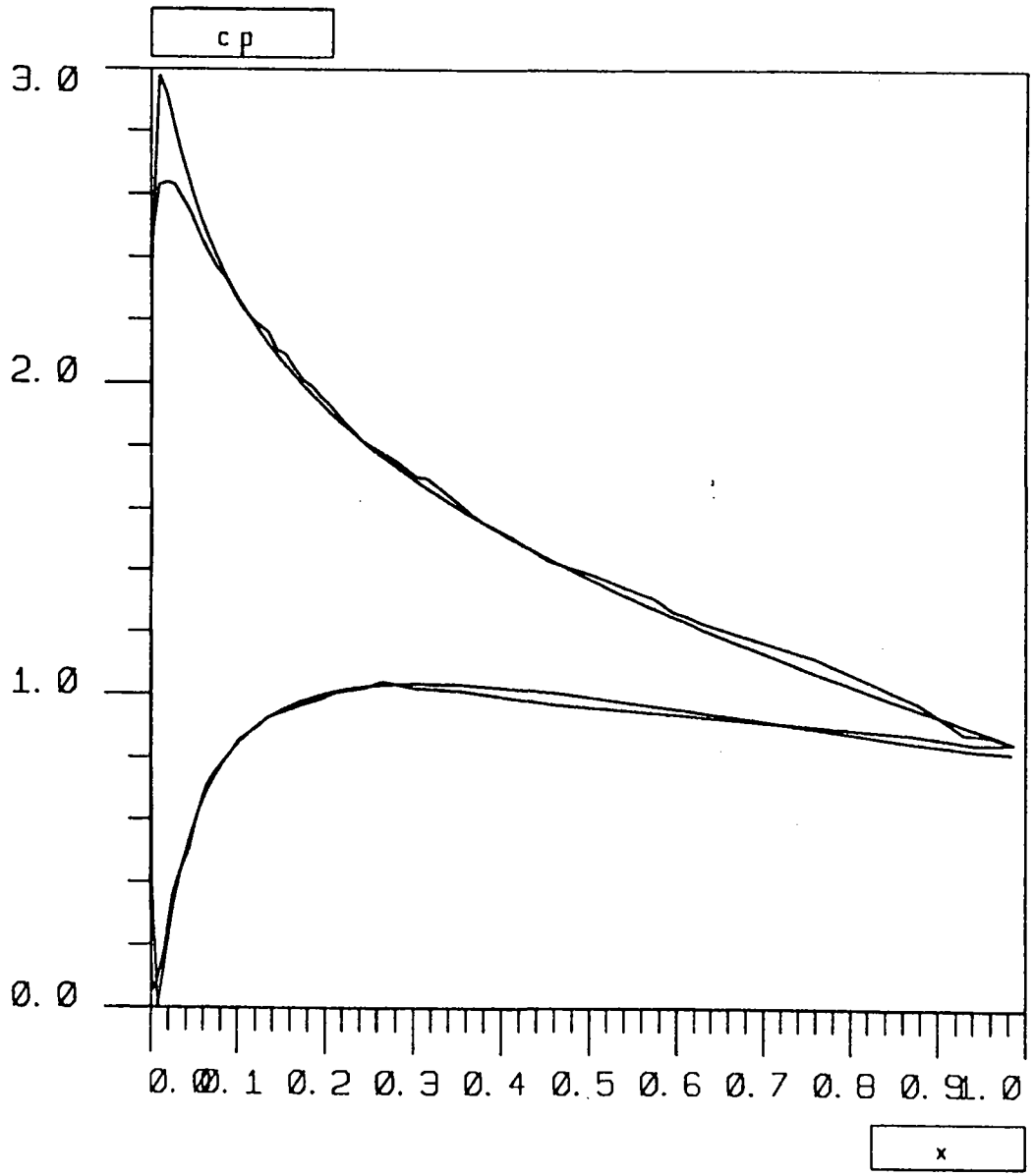
0.75E-16 0.99

EXTREMA EN Y :

0.43E-02 1.9

TRACE DE COURBES

Fig 3.7: Pressure coefficient on the airfoil versus x for the lifting case ($\alpha = 5^0$).



MODULEF : mvallet

alpha=5. ns=1545 tps=51 err=4.39%

18/11/91

keepp/jouk452.5.cp0

NOMBRE DE COURBES : 2

EXTREMA EN X :

0.75E-16 0.99

EXTREMA EN Y :

0.57E-02 3.0

TRACE DE COURBES

4. SOLUTION BY THE FINITE ELEMENT METHOD ON RECTANGULAR LOCALLY FITTED MESHES VIA FICTIOUS DOMAINS.

4.1 Variational formulations and approximations

We will consider problem (2.1)-(2.6) within the potential flow formulation (2.18)-(2.22) under the assumption that a streamline Σ is given. It follows from Section 2.2 that the solution function φ is a linear combination of the solution functions φ_1 and φ_2:

$$\varphi = \varphi_1 + \beta\varphi_2 \tag{4.1}$$

where

$$\begin{aligned} &\Delta\varphi_1 = 0 && \text{in } \Omega \\ &\frac{\partial\varphi_1}{\partial n} = u_\infty \cdot n && \text{on } \Gamma_\infty \\ &\frac{\partial\varphi_1}{\partial n} = 0 && \text{on } S \end{aligned} \tag{4.2}$$

and

$$\begin{aligned} &\Delta\varphi_2 = 0 \quad \text{in } \Omega \setminus \Sigma \\ &\frac{\partial\varphi_2}{\partial n} = 0 \quad \text{on } \Gamma_\infty \cup S \\ &\varphi_2 \,|_{\Sigma^+} = \varphi_2 \,|_{\Sigma^-} + 1. \end{aligned} \tag{4.3}$$

We replace (4.2) by:

$$\varphi_1 \in H^1(\Omega) : \int_\Omega \nabla\varphi_1 \cdot \nabla w \, d\Omega = \int_{\Gamma_\infty} (u_\infty \cdot n) w \, d\Omega \;\forall w \in H^1(\Omega) \tag{4.4}$$

and (4.3) by:

$$\varphi_2 \in H^1(\Omega \setminus \Sigma),\ \varphi_2 \,|_{\Sigma^+} = \varphi_2 \,|_{\Sigma^-} + 1 \ : \ \int_{\Omega\setminus\Sigma} \nabla\varphi_2 \cdot \nabla w \, d\Omega = 0 \;\forall w \in H^1(\Omega). \tag{4.5}$$

To solve (4.4) and (4.5) we apply the finite element method from Section 3. Thus we get

$$\varphi_1^h \in H_h^1(\Omega_h) : \int_{\Omega_h} \nabla\varphi_1^h \cdot \nabla w^h \, d\Omega = \int_{\Gamma_\infty} (u_\infty \cdot n) w^h \, d\Omega \;\; \forall w^h \in H_h^1(\Omega_h) \tag{4.6}$$

for problem (4.4) and

$$\varphi_2^h \in H_h^1(\Omega_h \setminus \Sigma),\ \varphi_2^h \,|_{\Sigma^+} = \varphi_2^h \,|_{\Sigma^-} + 1 \ : \ \int_{\Omega_h\setminus\Sigma} \nabla\varphi_2^h \cdot \nabla w^h \, d\Omega = 0 \;\; \forall w^h \in H_h^1(\Omega_h) \tag{4.7}$$

for problem (4.5).

Both FEM-problems (4.6) and (4.7) lead to the system of linear algebraic equations

$$A\Phi = f \tag{4.8}$$

with the same symmetric positive semidefinite $N \times N$-matrix A but with different right hand side vectors f. In both cases system (4.8) is compatible because $f \perp \ker A$.

The value of N is equal to the number of mesh vertices belonging to $\bar{\Omega}_h$.

4.2 Fictitious domain (fictitious components) approach

System (4.8) will be solved by a preconditioned conjugate gradient method (3.8)-(3.12) with the only difference that instead of C^{-1} we will use a symmetric positive semidefinite $N \times N$-matrix H with $\ker H = \ker A$, i.e. the null-spaces of H and A will coincide.

The construction of a preconditioner H is based on the fictitious domain idea with respect to algebraic problems [1, 7]. For this reason it was originally named as the fictitious components method [4, 5, 7]. Different kinds of preconditioners can be proposed. Here we will use separable preconditioners [6, 9]. To introduce separable preconditioners we have to use rectangular meshes for rectangular domains.

The fictitious domain method with separable preconditioners includes several stages:

1. embedding of an original domain Ω into a rectangle Π: $\Pi \supset \Omega$;
2. construction for Π of a rectangular mesh locally fitted to $\partial\Omega$;
3. definition of a separable matrix B related to the domain Π;
4. definition of a preconditioner H via the matrix B;
5. choice of the initial guess Φ^0 for the PCG-method (3.8)-(3.12) to operate within a subspace of h-harmonic finite element functions;
6. solving systems $Bv = g$ with sparse right-hand sides g, by a fast separable method.

The PCG-method is similar to Algorithm 1:

$$\Phi^0 = Hf, \tag{4.9}$$

$n = 0$:

$$\begin{aligned} \xi^0 &= A\Phi^0 - f, \\ g^0 &= h^0 = H\xi^0. \end{aligned} \tag{4.10}$$

1. Calculate:

$$\rho^n = \frac{(H\xi^n)^T \xi^n}{h^{nT} A h^n}, \tag{4.11}$$

$$\Phi^{n+1} = \Phi^n - \rho^n h^n, \tag{4.12}$$

$$\xi^{n+1} = \xi^n - \rho^n A h^n, \tag{4.13}$$

$$g^{n+1} = H\xi^{n+1}, \tag{4.14}$$

$$\gamma^n = \frac{(H\xi^{n+1})^T \xi^{n+1}}{(H\xi^n)^T \xi^n}, \tag{4.15}$$

$$h^{n+1} = g^{(n+1)} - \gamma^n h^n. \tag{4.16}$$

2. If $\| \xi^{n+1} \|_H < \varepsilon$, stop else increment n by 1 and go to 1.

4.3 Implementation

4.3.1 General description

Implementation of the fictitious domain method for our problem includes the following steps.

a). Choosing a rectangle $\Pi = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ such that $\Pi \supseteq \Omega$. We will suppose that $\Gamma_\infty = \partial\Pi$.

b). Constructing a rectangular grid $\Pi_h = \{(x_i, y_j),\ i = 1, ..., n_x,\ j = 1, ..., n_y\}$, where $x_1 < x_2 < ... < x_{n_x}$, $x_1 = x_{\min}$, $x_{n_x} = x_{\max}$, $y_1 < y_2 < ... < y_{n_y}$, $y_1 = y_{\min}$, $y_{n_y} = y_{\max}$. Let us denote

$$h^x_{i+1/2} = x_{i+1} - x_i,\ i = 1, ..., n_x - 1,$$
$$h^y_{j+1/2} = y_{j+1} - y_j,\ j = 1, ..., n_y - 1.$$

The couples $[(x_i, y_j), (x_{i+1}, y_j)]$ and $[(x_i, y_j), (x_i, y_{j+1})]$ will be refered as "grid edges" and the quadrangles with vertices (x_i, y_j), (x_{i+1}, y_j), (x_{i+1}, y_{j+1}), (x_i, y_{j+1}) as "grid cells".

c). Definition of the matrix B.

Let us consider problem (4.2) on the rectangle Π:

$$\begin{aligned} \Delta\varphi_1 &= 0 \qquad \text{in } \Pi \\ \frac{\partial\varphi_1}{\partial n} &= u_\infty \cdot n \quad \text{on } \partial\Pi. \end{aligned} \tag{4.17}$$

Let us triangulate the grid Π_h by means of cell diagonals and approximate problem (4.17) according to (4.4). We obtain the system of linear algebraic equations

$$B\Psi = \tilde{f} \tag{4.18}$$

with symmetric positive semidefinite $N_0 \times N_0$-matrix B where $N_0 = n_x n_y$. It has the following elements b_{kl}, $k = 1, ..., N_0$, $l = 1, ..., N_0$:

$$b_{kl} = \begin{cases} -\dfrac{h^x_{i_k-1/2} + h^x_{i_k+1/2}}{2h^y_{(j_k+j_l)/2}} & \text{if } i_k = i_l,\ 1 < i_k < n_x,\ |j_k - j_l| = 1; \\ -\dfrac{h^x_{3/2}}{2h^y_{(j_k+j_l)/2}} & \text{if } i_k = i_l = 1,\ |j_k - j_l| = 1; \\ -\dfrac{h^x_{n_x-1/2}}{2h^y_{(j_k+j_l)/2}} & \text{if } i_k = i_l = n_x,\ |j_k - j_l| = 1; \\ -\dfrac{h^y_{j_k-1/2} + h^y_{j_k+1/2}}{2h^x_{(i_k+i_l)/2}} & \text{if } j_k = j_l,\ 1 < j_k < n_y,\ |i_k - i_l| = 1; \\ -\dfrac{h^y_{3/2}}{2h^x_{(i_k+i_l)/2}} & \text{if } j_k = j_l = 1,\ |i_k - i_l| = 1; \\ -\dfrac{h^y_{n_y-1/2}}{2h^x_{(i_k+i_l)/2}} & \text{if } j_k = j_l = n_y,\ |i_k - i_l| = 1; \\ 0 & \text{otherwise if } k \neq l; \end{cases} \tag{4.19}$$

$$b_{kk} = -\sum_{\substack{l=1 \\ l \neq k}}^{N_0} b_{kl}.$$

Here, some ordering (i_k, j_k), $k = 1, ..., N_0$, of vertices of Π_h has been used.

d). Construction of the grid $\bar{\Omega}_h$ by means of the local fitting of Π_h to S will be considered in Subsection 4.3.2.

e). Definition of a preconditioner H via the matrix B will be considered in Subsection 4.3.3.

4.3.2 Mesh generator

In the present work a locally fitted grid was constructed by means of the following algorithm.

Step 1. Starting with the point P (see Fig. 2.1) we trace the curve S clockwise and find a sequence $d_1 d_2 ... d_m$ of grid edges and vertices which are succesively crossed by this curve ($d_1 = P$).

Step 2. Construction of a sequence $M = P_1 P_2 ... P_n$ of grid vertices to be modified later.

Initially we set $P_1 = P$ (see Fig. 5.1) and then we succesively consider the elements d_j, $j = 2, ..., m$. Let $P_1 ... P_t$, $t \leq n$, be the points already included into M for a given j. Two cases are possible:

1) d_j is a vertex of Π_h.

In this case we add it to M ($P_{t+1} = d_j$) iff it has not yet been included.

2) d_j is an edge of Π_h with ending points $P^{(1)}, P^{(2)}$.

If $P^{(1)} = P_t$ or $P^{(2)} = P_t$ then we don't add any point to M for the given j. Otherwise we include one of the points $P^{(1)}, P^{(2)}$ into M as P_{t+1}. If $P^{(1)} = P_i$ for

some $i < t$ then we add to M the point $P^{(2)}$, and vice versa. If both points are not yet included into M then we add the point nearest to S.

Step 3. Modification of the vertices $P_1, P_2, ..., P_n$.

For each grid vertex $P_i \in M$ we find a point $\hat{P}_i \in S$ nearest to P_i and then substitute $\hat{P}_i$ for P_i.

Let us denote the new grid by $\hat{\Pi}_h$ and also $S_h = \hat{P}_1\hat{P}_2...\hat{P}_n\hat{P}_1$. This broken line approximates the curve S (the bold line in Fig. 5.1).

Step 4. Triangulation of modified cells.

At this step we triangulate the grid cells of $\hat{\Pi}_h$ which have at least one modified vertex $\hat{P}_i$, $i \leq n : \hat{P}_i \neq P_i$. Each cell is divided into two triangles by drawing a diagonal. If one of diagonals of a cell belongs to S_h then we choose it for the triangulation. Otherwise we choose the shortest diagonal.

Step 5. Construction of Ω_h.

At this step we construct a grid Ω_h for the finite element approximationt to the problem under consideration. We include into Ω_h every cell of $\hat{\Pi}_h$ belonging to Ω and also the triangles whose barycenters belong to Ω.

The result of this algorithm can be seen on Fig. 5.1 for the particular profile S.

4.3.3 Definition of a matrix H

Let us represent the matrix B in the block form:

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \tag{4.20}$$

where B_{11} is an $N \times N$ matrix and the same vertex ordering is used for A and B_{11}. Hereafter we will identify the N_0-dimensional vectors and the grid functions on $\hat{\Pi}_h$ using the vertex ordering of (4.20).

Let us introduce the Schur complement matrix

$$B_S = B_{11} - B_{12}B_{22}^{-1}B_{21} \tag{4.21}$$

and define H as follows:

$$H = B_S^+, \tag{4.22}$$

where the symbol "+" denotes the generalized inverse of a matrix.

To multiply some vector ξ by the matrix H it is necessary to solve the following system of equations:

$$B \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \xi \\ 0 \end{bmatrix}. \tag{4.23}$$

It is easy to see that

$$H\xi = v_1. \tag{4.24}$$

System (4.23) can be solved by means of the well-known fast direct method that uses separation of variables (see Subsection 4.3.5).

4.3.4 Implementation within the subspace of h-harmonic functions

Let us introduce the matrix

$$C = B_S - A. \tag{4.25}$$

It is easy to see that all the rows of B_S and A coincide except for the rows corresponding to vertices $\hat{P}_i \in S_h$ (marked by squares in Fig. 5.1) and neighboring vertices of Ω_h (marked by circles in Fig. 5.1). We denote by $\hat{S}_h$ the latter set of vertices. Thus, all the rows of C are equal to zero except for the rows corresponding to vertices of $S_h \cup \hat{S}_h$. The total number of such vertices is of the order of $N^{1/2}$.

It follows from (4.9), (4.10), (4.22) and (4.25) that

$$\xi^0 = -CHf. \tag{4.26}$$

Hence, $\xi^0 \in \operatorname{im} C$, i.e. this residual vector is equal to zero everywhere except on the vertices of $S_h \cup \hat{S}_h$. Let us also consider the vectors Ag^0 and Ah^0 (see (4.10):

$$Ag^0 = Ah^0 = \xi^0 - CH\xi^0. \tag{4.27}$$

These two vectors are also equal to zero everywhere except on the vertices of $S_h \cup \hat{S}_h$.

Method (4.11)-(4.16) can be rewritten as follows:

$$\rho^n = \frac{(H\xi^n)^T \xi^n}{h^{nT} A h^n}, \tag{4.28}$$

$$C\Phi^{n+1} = C\Phi^n - \rho^n C h^n, \tag{4.29}$$

$$\xi^{n+1} = \xi^n - \rho^n A h^n, \tag{4.30}$$

$$Ag^{n+1} = \xi^{n+1} - CH\xi^{n+1}, \tag{4.31}$$

$$\gamma^n = \frac{(H\xi^{n+1})^T \xi^{n+1}}{(H\xi^n)^T \xi^n}, \tag{4.32}$$

$$Ah^{n+1} = Ag^{(n+1)} - \gamma^n A h^n. \tag{4.33}$$

It is easy to see from (4.29)-(4.31), (4.33), that all the vectors ξ^n, Ag^n, Ah^n (and, of course, Cg^n, $C\Phi^n$) belong to the image of C, i.e. are equal to zero everywhere except on the vertices of $S_h \cup \hat{S}_h$. In other words, all the grid functions u_h^n corresponding to Φ^n from (4.12) and all the errors $\psi^n = u_h^n - u_h$ are h-harmonic grid functions everywhere except on the vertices of $S_h \cup \hat{S}_h$.

Thus, we can implement this iterative process by storing only $O(N^{1/2})$ components of vectors ξ^n, Ag^n, Ah^n, Cg^n, $C\Phi^n$ (because the other components are equal to zero). Calculation of linear combinations of vectors in (4.27), (4.29)-(4.31), (4.33) requires $O(N^{1/2})$ arithmetic operations.

Let us consider the calculation of a vector $CH\xi$ which is necessary for implemention of (4.26), (4.27) and (4.31). It follows from (4.20)-(4.25) that

$$CH\xi = Cv_1 = (B_{11} - A)v_1 + B_{12}v_2. \tag{4.34}$$

It is easy to see from (4.34) that to calculate vector $CH\xi$ we need values of v_1 only at vertices of $S_h \cup \hat{S}_h$ (marked by circles and squares in Fig. 5.1) and also we need values of v_2 at vertices of $\Pi_h \setminus \bar{\Omega}_h$ neighboring to S_h (marked by small triangles in Fig. 5.1). We denote by $\tilde{S}_h$ the

latter set of vertices. A total number of vertices of $\tilde{S}_h$ is also of the order of $N^{1/2}$. Thus, while a vector v from (4.23) is known the calculation of $CH\xi$ requires $O(N^{1/2})$ arithmetic operations.

To calculate the scalar products from (4.28), (4.32) it is sufficient to know vectors $H\xi^n$, h^n only at vertices of $S_h \cup \hat{S}_h$ because the second multiplier, ξ^n or Ah^n, is not equal to zero only at these vertices. Calculation of these scalar products requires $O(N^{1/2})$ arithmetic operations as well.

It follows from above that the implementation of the method under consideration requires $O(N^{1/2})$ computer memory locations and $O(N^{1/2})$ arithmetic operations except for solution of problem (4.23).

4.3.5 Partial solution algorithm

To implement the method considered the linear system of equations

$$Bv = \xi, \tag{4.35}$$

should be solved at each step, where a right-hand side ξ belongs to the subspace $\operatorname{im} C$, i.e. it can be distinct from zero only at vertices of $S_h \cup \hat{S}_h$. It is necessary to calculate a solution v only at vertices of $S_h \cup \hat{S}_h \cup \tilde{S}_h$. This problem is called "the partial solution problem".

Now we will describe an algorithm for solution of this problem which was proposed in [2, 4]. It is a modification of the well-known fast direct method that uses separation of variables.

System (4.35) can be rewritten as follows (see (4.19)):

$$\begin{gathered}
(\frac{1}{h^x_{3/2}} I + \frac{h^x_{3/2}}{2} D^{-1}T)v_1 - \frac{1}{h^x_{3/2}} v_2 = D^{-1}\xi_1, \\
-\frac{1}{h^x_{i-1/2}} v_{i-1} + ((\frac{1}{h^x_{i-1/2}} + \frac{1}{h^x_{i+1/2}})I + \frac{h^x_{i-1/2} + h^x_{i+1/2}}{2} D^{-1}T)v_i - \frac{1}{h^x_{i+1/2}} v_{i+1} = D^{-1}\xi_i, \\
i = 2, ..., n_x - 1, \\
-\frac{1}{h^x_{n_x-1/2}} v_{n_x-1} + (\frac{1}{h^x_{n_x-1/2}} I + \frac{h^x_{n_x-1/2}}{2} D^{-1}T)v_{n_x} = D^{-1}\xi_{n_x},
\end{gathered} \tag{4.36}$$

where v_i, $i = 1, ..., n_x$, is a n_y-vector of values on a grid line $x = x_i$; ξ_i, $i = 1, ..., n_x$, is the same vector for the right-hand side ξ; T is the following symmetric tridiagonal $n_y \times n_y$-matrix:

$$T = \begin{bmatrix}
\frac{1}{h^y_{3/2}} & -\frac{1}{h^y_{3/2}} & 0 & \cdots & 0 \\
\ddots & \ddots & \ddots & \ddots & \vdots \\
0 & -\frac{1}{h^y_{j-1/2}} & \frac{1}{h^y_{j-1/2}} + \frac{1}{h^y_{j+1/2}} & -\frac{1}{h^y_{j+1/2}} & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & -\frac{1}{h^y_{n_y-1/2}} & \frac{1}{h^y_{n_y-1/2}}
\end{bmatrix}, \tag{4.37}$$

D is the following diagonal $n_y \times n_y$-matrix:

$$D = \begin{bmatrix} \frac{h^y_{3/2}}{2} & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \frac{h^y_{j-1/2} + h^y_{j+1/2}}{2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \frac{h^y_{n_y-1/2}}{2} \end{bmatrix} \tag{4.38}$$

and I is the $n_y \times n_y$ identity matrix.

Let us consider the following eigenvalue problem

$$D^{-1}T\omega = \lambda\omega, \tag{4.39}$$

which is equivalent to the generalized symmetric eigenvalue problem

$$T\omega = \lambda D\omega. \tag{4.40}$$

Let us denote the eigenpairs of (4.39) by $\{\lambda_l, \omega^l\}$, $l = 1, ..., n_y$,. It is well known that the set of vectors ω^l, $l = 1, ..., n_y$, is a D-orthogonal basis in R^{n_y}. We will suppose that it is a D-orthonormalised basis:

$$(D\omega^k, \omega^l) = \begin{cases} 1 & \text{if } k = l; \\ 0 & \text{otherwise.} \end{cases} \tag{4.41}$$

We will try the solution to (4.36) in the form:

$$v_i = \sum_{l=1}^{n_y} v_i^l \omega^l, \ i = 1, ..., n_x, \tag{4.42}$$

where v_i^l, $l = 1, ..., n_y$, are some coefficients. Let us represent a right-hand side of (4.36) in the same form:

$$D^{-1}\xi_i = \sum_{l=1}^{n_y} \xi_i^l \omega^l, \ i = 1, ..., n_x. \tag{4.43}$$

It follows from (4.41) that

$$\xi_i^l = (\omega^l, \xi_i) = \sum_{j=1}^{n_y} \xi_{ij}\omega_j^l, \ i = 1, ..., n_x, \ l = 1, ..., n_y. \tag{4.44}$$

Substituting (4.42) and (4.43) into (4.36) we obtain n_y independent linear systems of

equations with the tridiagonal $n_x \times n_x$-matrices:

$$\begin{cases} (\frac{1}{h^x_{3/2}} + \frac{h^x_{3/2}}{2}\lambda_l)v^l_1 - \frac{1}{h^x_{3/2}}v^l_2 = \xi^l_1, \\ -\frac{1}{h^x_{i-1/2}}v^l_{i-1} + ((\frac{1}{h^x_{i-1/2}} + \frac{1}{h^x_{i+1/2}}) + \frac{h^x_{i-1/2} + h^x_{i+1/2}}{2}\lambda_l)v^l_i - \frac{1}{h^x_{i+1/2}}v^l_{i+1} = \xi^l_i, \\ i = 2, ..., n_x - 1, \\ -\frac{1}{h^x_{n_x-1/2}}v^l_{n_x-1} + (\frac{1}{h^x_{n_x-1/2}} + \frac{h^x_{n_x-1/2}}{2}\lambda_l)v^l_{n_x} = \xi^l_{n_x}, \\ l = 1, ..., n_y. \end{cases} \tag{4.45}$$

Thus, we have the following algorithm for solving (4.35):

Algorithm. For $l = 1, 2, ..., n_y$:

1. Calculate ξ^l_i, $i = 1, ..., n_x$, for a given l by means of formula (4.44).
2. Solve system (4.45) for a given l by means of the Gauss elimination method for tridiagonal matrices, i.e. we calculate the coefficients v^l_i.
3. Add the lth items in (4.42) only for vertices of $S_h \cup \hat{S}_h \cup \tilde{S}_h$.

Let us estimate the number of arithmetic operations in this Algorithm. The first step requires $O(N^{1/2})$ operations for each l because ξ_{ij} can be nonzero only at vertices $(x_i, y_j) \in S_h \cup \hat{S}_h$ and $\dim S_h \cup \hat{S}_h = O(N^{1/2})$. The third step requires $O(N^{1/2})$ operations for each l as well because we calculate v_{ij} only at vertices $(x_i, y_j) \in S_h \cup \hat{S}_h \cup \tilde{S}_h$ and $\dim S_h \cup \hat{S}_h \cup \tilde{S}_h = O(N^{1/2})$. The second step obviously requires $O(n_x)$ operations for each l. Hence, partial solution of problem (4.35) requires $O(N^{1/2} + n_x) \cdot n_y = O(N_0)$ arithmetic operations.

It is easy to see that it requires $O(N^{1/2})$ computer memory locations.

It follows from the present and previous subsections that the method under consideration requires $O(N^{1/2})$ computer memory locations and each step requires $O(N_0)$ arithmetic operations.

Remark. To use the above Algorithm we have to know the eigenpairs $\{\lambda_l, \omega^l\}$, $l = 1, ..., n_y$, of problem (4.39). They are known in an explicit form if the grid y_j, $j = 1, ..., n_y$, is uniform. Otherwise they can be calculated using QR-algorithm and method of inverse iterations. It requires $O(n_y^2) = O(N_0)$ arithmetic operations as well, because T is a tridiagonal matrix and D is a diagonal matrix.

4.3.6 Calculation of an approximate solution

After completion of the iterative method (4.26)-(4.33) we do not yet have the approximate iterative solution $\tilde{\Phi}$ itself. We only have the vector $C\Phi^{n_{max}}$. To calculate $\tilde{\Phi}$ we use the following obvious relation for the precise solution Φ:

$$B_S\Phi = C\Phi + f. \tag{4.46}$$

Substituting $C\Phi^{n_{max}}$ by $C\Phi$ in (4.46) we obtain

$$\tilde{\Phi} = H(C\Phi^{n_{max}} + f) = HC\Phi^{n_{max}} + \Phi^0. \tag{4.47}$$

Only some components of $\tilde{\Phi}$ are usually needed in practice. To calculate them we can solve the corresponding partial solution problem (4.35) with the right-hand side $C\Phi^{n_{max}} + f$.

4.3.7 Generalization

It is easy to see that the same method can be used for the stream function approach (2.8)-(2.17). In this case a vector $\xi \in \mathrm{im}\, C$ can be nonzero only at the vertices of $\hat{S}_h$ (marked by circles in Fig. 5.1). To calculate a vector $CH\xi$ we need the values of v_1 (see (4.23)-(4.24)) only at the vertices of $\hat{S}_h$ and also we need values of v_2 at vertices of S_h (marked by squares in Fig. 5.1).

4.4 Numerical experiments

In this Section we present the numerical results for the Joukowski airfoil described in Section 2.3 solved by means of the fictitious domain method.

We chose $\Pi = \{(x,y) : x \in [-5,11],\ y \in [-5,5]\}$, $\Gamma_\infty = \partial\Pi$, $|u_\infty| = 1$, and placed the profile S so that the leading edge has the coordinates $(0,0)$ and the trailing edge P - the coordinates $(1,0)$.

The rectangular grid Π_h was constructed as follows. First, we chose some basic minimum stepsize $h_{\min}$. Then we constructed a grid $x_{n_x^0}$, $x_{n_x^0+1}, ..., x_{n_x^0+n_x^s}$ on the segment $[0,1]$ of the X-axis as follows:

$$x_{n_x^0} = 0,$$
$$x_{n_x^0+1} - x_{n_x^0} = h_{\min},$$
$$\frac{x_{i+1} - x_i}{x_i - x_{i-1}} = q_1 = \mathrm{const},\ \ i = n_x^0 + 2, ..., n_x^0 + n_x^s - 1,$$

where the value of n_x^s was chosen "by hand". Besides, we constructed the uniform mesh with the stepsize $h_{\min}$ on a segment $[-\tilde{t}_{\max}, \tilde{t}_{\max}]$ of the Y-axis where $\tilde{t}_{\max} = n_y^s h_{\min}$, $n_y^s = 2([\frac{t_{\max}}{2h_{\min}}] + 1)$, $t_{\max}$ is a maximum thickness of the profile S [3]:

$$t_{\max} = \frac{3\sqrt{3}}{4}\epsilon.$$

Outside the rectangle $\{(x,y) : x \in [0,1],\ y \in [-\tilde{t}_{\max}, \tilde{t}_{\max}]\}$ we constructed a mesh with a stepsize exponentially increasing (moving away from S) along each of the axes X and Y, i.e. the ratio of two neighboring steps was equal to $q_2 > 1$.

The grid shown in Fig. 5.1 corresponds to the following values of these parameters:

$$h_{\min} = 0.01,\ n_x^s = 40,\ q_2 = 1.3.$$

Then

$$n_x = 76,\ n_y = 56,\ N_0 = 4256.$$

The calculations were carried out on this grid and also on the grid with the parameters:

$$h_{\min} = 0.005,\ n_x^s = 50,\ q_2 = 1.3,$$
$$n_x = 88,\ n_y = 76,\ N_0 = 6688.$$

In the experiments we used the potential function approach described in Section 2.1 as well as the stream function approach described in Section 2.2. In each calculation we solved two linear system of equations with the same matrix. It is easy to see that for zero angle of attack it is sufficient to solve only one system of equations because $\beta = 0$ (see (4.1)). Besides, the function φ_2 in (4.1) doesn't depend on the angle of attack α, hence, it is sufficient to find it only once in a set of calculations for several values of α on one and the same grid. But we didn't use these possibilities intentionally.

The criterion ε from Section 4.2 was chosen empirically so that the precision (2.30) should not considerably change for $n > n_{\max}$.

The results of calculations for the stream function approach are presented in Table 5.1. The results of calculations for the potential function approach are presented in Table 5.2. In the third column of these tables the number of iterations needed for solving both linear systems with the precision ε are shown. Computing time is given for the computer Apollo-DN4000. It includes time for:

1) grid and problem generation ($\sim 10\%$ of time);

2) solution of the eigenvalue problem (4.39) ($\sim 10\%$ of time);

3) solution of two linear systems of equations by means of the iterative process (4.26)-(4.33) ($\sim 80\%$ of time);

4) postprocessing (4.47) ($\sim 2\%$ of time).

It doesn't include time required for plotting.

The functions $|u_h^{stream}|^2 \,|_S$, $|u_h^{potential}|^2 \,|_S$ as well as the exact solution $|u^{exact}|^2 \,|_S$ (see Subsection 2.3) are presented in Figure 5.2 (for $\alpha = 5°$).

Analysis of Tables 5.1-5.2 and Figure 5.2 shows that the solution of the problem considered by means of the potential function approach takes a little more time than by means of the stream function approach (on the same grid). Nevertheless, the potential function approach requires sufficiently less computing time for obtaining a given precision (2.30). Thus, it is preferable to use the latter.

It should also be noted that the value of ε in the stream function approach must be greater than in the potential function one.

The first grid

$h_{\min} = 0.01,\ n_x^s = 40,\ q_2 = 1.3,\ n_x = 76,\ n_y = 56,\ N_0 = 4256$

Angle of attack α	ε	Number of iterations	Computing time	Precision (2.30) (in %)
0°	10^{-4}	20	17"	3 %
2°	10^{-4}	19	16"	3.9 %
5°	10^{-5}	24	19"	7.3 %
10°	10^{-5}	23	18"	15.5 %

The second grid

$h_{\min} = 0.005,\ n_x^s = 50,\ q_2 = 1.3,\ n_x = 88,\ n_y = 76,\ N_0 = 6688$

Angle of attack α	ε	Number of iterations	Computing time	Precision (2.30) (in %)
0°	10^{-5}	30	36"	2 %
2°	10^{-6}	37	43"	2.3 %
5°	10^{-6}	34	40"	4 %
10°	10^{-6}	34	40"	8.3 %

Table 4.1. Results of calculations for the stream function approach

The first grid

$h_{\min} = 0.01,\ n_x^s = 40,\ q_2 = 1.3,\ n_x = 76,\ n_y = 56,\ N_0 = 4256$

Angle of attack α	ε	Number of iterations	Computing time	Precision (2.30) (in %)
0°	10^{-3}	22	19"	0.9 %
2°	10^{-3}	30	24"	1.1 %
5°	10^{-3}	31	25"	2.1 %
10°	10^{-3}	32	25"	4.4 %

The second grid

$h_{\min} = 0.005,\ n_x^s = 50,\ q_2 = 1.3,\ n_x = 88,\ n_y = 76,\ N_0 = 6688$

Angle of attack α	ε	Number of iterations	Computing time	Precision (2.30) (in %)
0°	10^{-3}	24	33"	0.5 %
2°	10^{-3}	36	45"	0.6 %
5°	10^{-3}	37	46"	0.9 %
10°	10^{-3}	38	47"	1.7 %

Table 4.2. Results of calculations for the potential function approach

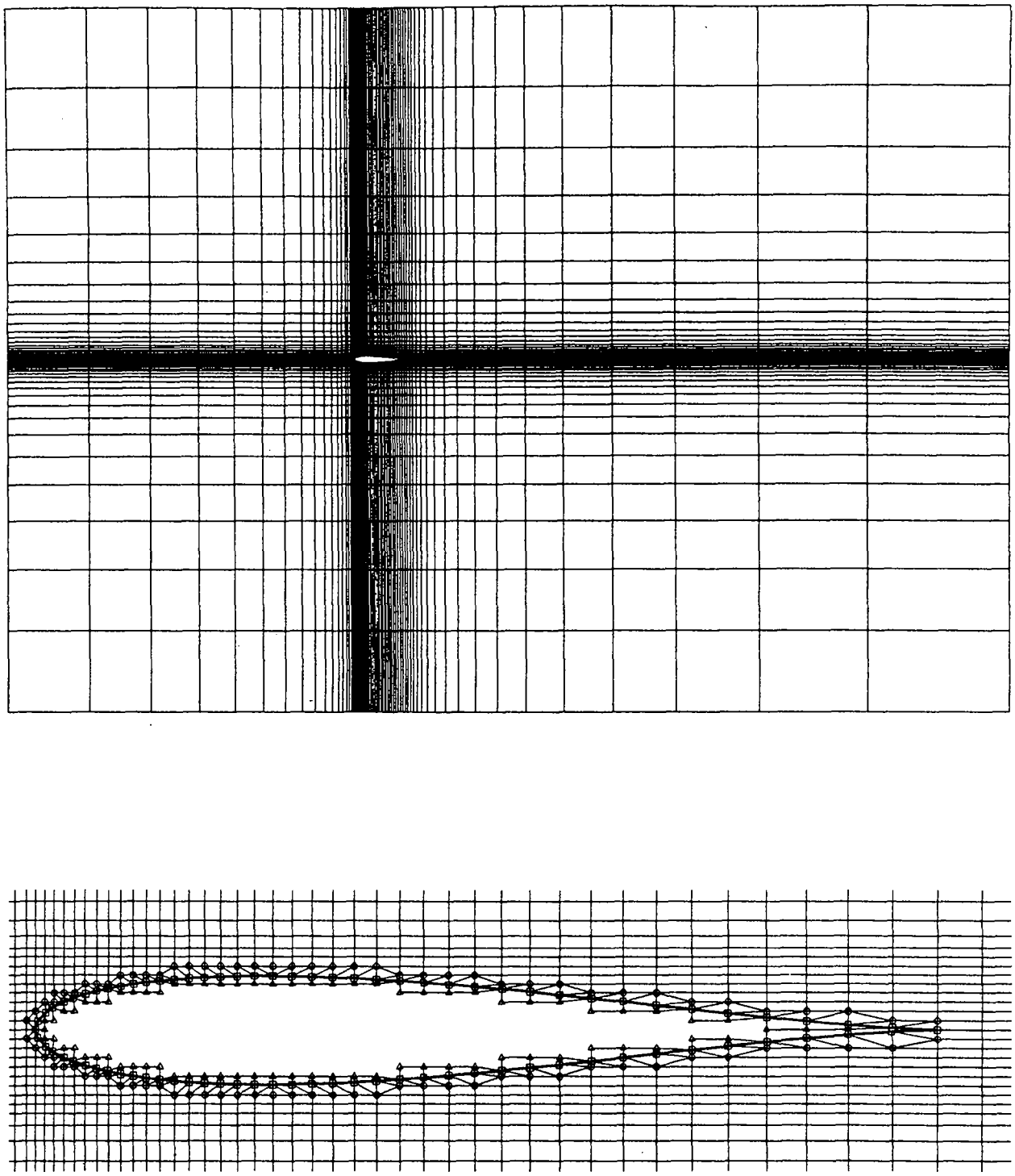
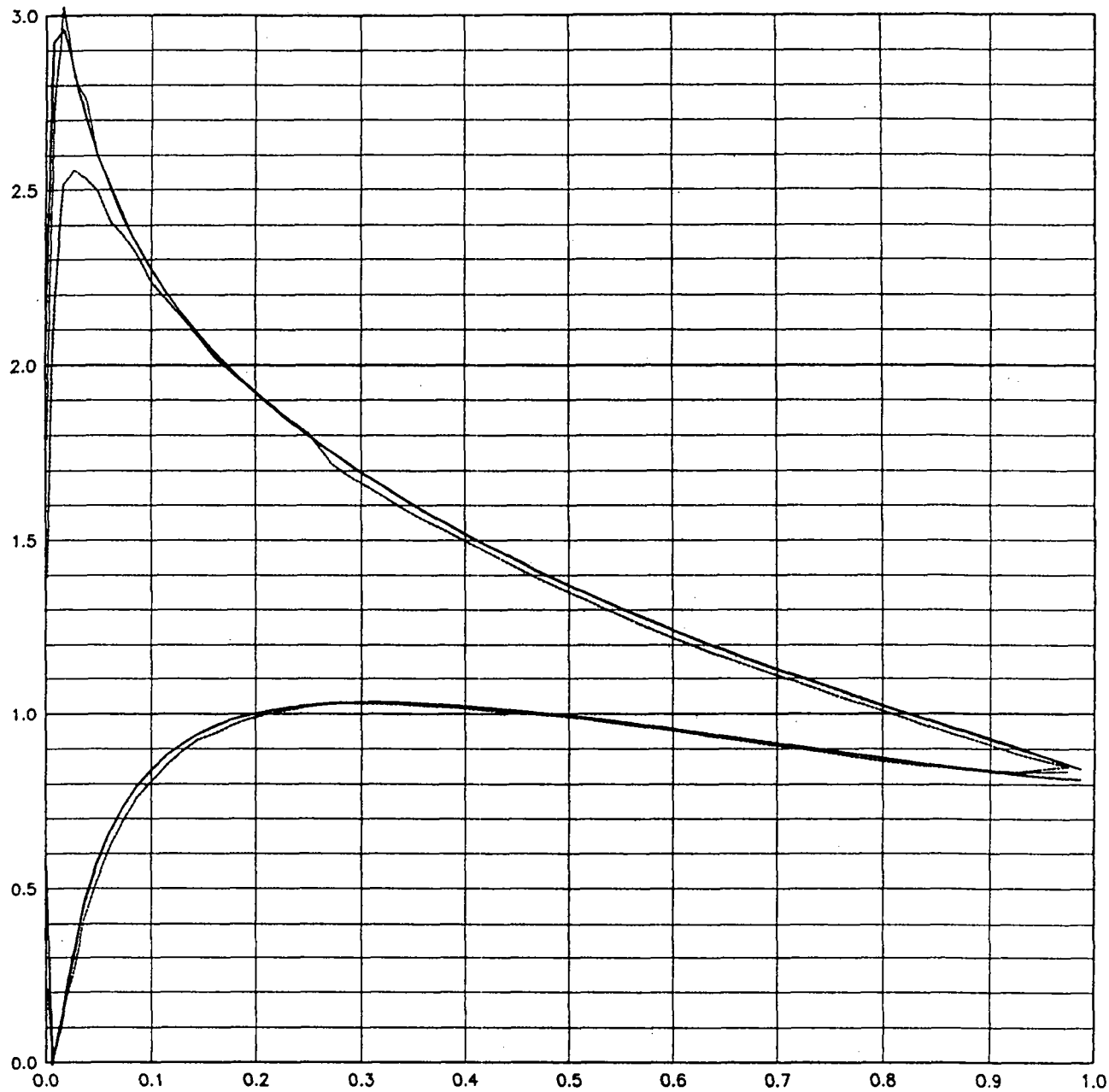


Figure 4.1. The locally fitted grid for the problem considered



The solid line - $|u^{exact}|^2\,|_S$

The dashed line - $|u_h^{stream}|^2\,|_S$

The dotted line - $|u_h^{potential}|^2\,|_S$

Figure 4.2. Solution on the Joukowski profile S

References

[1] G.P. Astrakhantsev
Iterative methods for solving variational difference schemes for two dimensional second order elliptic equations
Ph. D. thesis (in Russian). LOMI Akad. Nauk SSSR, Leningrad, 1972

[2] A. Banegas
Fast Poisson solvers for problems with sparsity
Math. Comp., pp.441-446, No 3, 1978

[3] K. Krishnamurty
Principle of ideal fluid aerodynamics
John Wiley, 1966

[4] Yu.A. Kuznetsov
Matrix computational processes in subspaces
Comp. Math. in Appl. Sci. and Eng. VI (Eds. R. Glowinski and J.-L. Lions), pp.15-31, North Holland, Amsterdam, 1984

[5] Yu.A. Kuznetsov
Matrix iterative methods in subspaces
Proc. Int. Congress Math., Warsaw, 1983. pp.1509-1521, North Holland, Amsterdam, 1984

[6] Yu.A. Kuznetsov
Numerical methods in subspaces
Vychislitel'nye Protsessy i Sistemy (Computational Processes and Systems) (in Russian), pp.265-350, vol.2, Nauka, Moscow, 1985

[7] G.I. Marchuk, Yu.A. Kuznetsov
Some aspects of iterative methods
Vychislitel'nye Metody Lineinoy Algebry (Computational Methods of Linear Algebra) (Ed. G.I. Marchuk) (in Russian), pp.4-20, Vychisl. Tsentr Sib. Otdel. Akad. Nauk SSSR, Novosibirsk, 1972

[8] G.I. Marchuk, Yu.A. Kuznetsov, A.M. Matsokin
Fictitious domain and domain decomposition methods
Sov. J. Numer. Anal. Math. Modelling, pp.3-35, Vol.1, 1986

[9] W. Proskurowski, O.B. Widlund
On the numerical solution of Helmholtz's equation by the capacitance matrix method
Math. Comp., pp.433-468, No 30, 1976

[10] P. Grisvard:
Elliptic problems in non-smooth domains . Pitman, 1985.

[11] J.A. Meijerink- H.A. Van der Vorst:
An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. of Comp* . **31**, 148-162 ,1977.

[12] R. Sibson
Comput. Journal, vol. 21, pp. 243-245, 1978.

[13] C. L. Lawson
"Generation of a Triangular Grid with Application to Contour Plotting", *Cal. Tech. Jet Propulsion Lab. Tech Memorandum 299*, 1972.

[14] J. Peraire, M. Vahdati, K. Morgan, O. C. Zienkiewicz
Adaptive Remeshing for Compressible Flow Computations, *Journal of Computational Physic, vol 72, pp. 449-466*, 1987.

[15] E. F. D'Azevedo, R. B. Simpson
On Optimal Triangular Meshes For Minimizing the Gradient Error; *Numerische Mathematik*, to appear, 1991.

[16] F. Hecht, E. Saltel
EMC2 un logiciel d'edition de maillage et de contour bi-dimensionnel.*Rapport technique INRIA* 118. 1990.

[17] D. Young, R. Melvin, F. Johnson, J. Bussoletti, L. Wigton, S. Samant.
Application of sparse matrix solvers as effective preconditioners.*SIAM J. Sci. Stat. Comput.* (10)6,p118-1199, 1989.

[18] EM-TRANAIR
A computer program for the solution of Maxwell equations. AFFDL-TR-87 -3082. Flight Dynamics Lab. Wright Patterson Air Force Base. Dec 1987.

[19] R. Bank
The efficient implementation of local mesh refinement algorithms. *Adaptive Computational Methods.* (I. Babuska ed) SIAM, Philadelphia, 1983.

[20] R.Verfurth.
A Posteriori Error Estimators And Adaptive Mesh Refinements for the Navier Stokes Equations. In Gunzburger, M. D. Nicolaides, R. A. (Eds) *Incompressible CFD- Trends and Advances.* Cambridge University Press 1992.

[21] P.L. Georges, F. Hecht, M.G. Vallet
Determination of internal points for Voronoi automatic mesh generators. In *Advances in Engineering Software special issue on Mesh Generation* To appear.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249-6399