A numerical calculation of the flow through the actuator strip

van Helden, W.G.J.

Award date:
1988

A NUMERICAL CALCULATION OF THE FLOW THROUGH THE ACTUATOR STRIP

Wim van Helden

March, 1988 R-898-A

Thesis of the graduationwork, accomplished between
April, 1987 and March, 1988.
W.G.J. van Helden.

coaches: Gijs van Kuik
Paul Smulders.

Faculty of Physics, Fluid Dynamics, Wind Energy Group,
Eindhoven University of Technology.

'Rommelig,' besloot mijnheer Charles C.M. Carlier. 'En slordig!' Het was, als je 't goed naging, eigenlijk allemaal nogal rommelig, slordig eigenlijk, en onafgewerkt; - de schepping en zo, de sterrenhemel en zo, zelfs al was het de zomerse sterrenhemel van zijn geliefd Provence.

Havank: Lijk Halfstok.

Dit afstudeer verslag draag ik op aan mijn ouders.

Contents.

Summary.

The actuator strip is a massless strip of infinite length which is covered with a uniform force distribution. It is used as a two-dimensional model for the flow through propellors or windturbines.
Two different computer programmes have been developed; one for the calculation of the unsteady flow through the actuator strip with only one edge (the half-infinite actuator strip), the other for the calculation of the unsteady flow through the finite actuator strip.
In both cases, the flow at a particular time is found by solving the vorticity diffusion equation and the Poisson-PSI equation for the streamfunction.
The results for the half-infinite strip are good. The strip, switched on at $t = 0$, produces a velocity field with concentric, circular streamlines. The edge of the strip is the centre of the circles. In a great part of the domain the velocity field resembles the velocity field of a point vortex.
Also a comparison is made between the calculations and an analytical solution.
For the calculation of the flow through the finite actuator strip, a form of adaptive grid generation has been used. At each time step in the calculation the grid points are redistributed dependent on the solution of the differential equations in the previous time step. The adaptive grid generation has been made as simple as possible. In this way the principles of grid generation have become much clearer.
Because of the complexity of the computer programme it wasn't possible to finish the work completely. Therefore only preliminary results concerning the computed flow through the finite actuator strip are given.

The importance of windturbines in electricity generation or water pumping is still increasing, while helicopters and propellor-driven aircraft already play a very important role in air traffic and air transport. All three types of machines are rotating lift machines, they exchange energy with the surrounding air by revolving several rotorblades.
The goal of the designers of for instance windturbines is to make a windturbine with maximum power output at minimum cost. In order to obtain this goal it is necessary to have a good theory of the flow through a rotor. With such a theory one should be able to make an accurate prediction of the maximum power that can be extracted from the wind by a rotor before the rotor is built.
The ratio of the extracted power to the total power of the undisturbed wind going through the area of the rotor is called C_p. The value of C_p is given by the theory of Betz & Lancester and has a maximum of $\approx$ 59 %.
There are some indications, however, that this Lancester maximum and therefore this theory needs some modification.[ref.12]
The rotating movement of the rotor aggravates a straightforward theoretical modeling of the problem. For this reason some simplifications are used.
An example of a commonly used model is that of the actuator disc. This is a disc without mass that is covered with a force distribution. An imaginable example for a braking actuator disc is a circular screen or wire gauze.
This actuator disc model is often used to calculate the pattern of the flow through a rotor and especially the formation of vortices at the edge of the disc.
One suggestion is that discrete forces resulting from a flow singularity at the edge of the disc, cause a higher C_p. [ref.12,ref.6].
The problem of calculating the flow through an actuator disc can be tackled in different ways. For instance, Alex van der Spek [ref.13] gives a first order solution of Wu's equations [ref.14].
In this work, the 2-dimensional analogy of the actuator disc, the actuator strip, is used. The flow through the actuator strip is only a function of the x- and y-coordinates and thus the vorticity has only a

z-component. Therefore, only the component for the z-direction of the vorticity diffusion equation has to be solved to obtain the vorticity. Three relations, the vorticity diffusion equation, the Poisson-PSI equation to calculate the streamfunction from the vorticity, and the relation between velocity and streamfunction, give a closed set of equations to determine the flow as a function of time.

The aim of this work is to get insight into the behaviour of the flow and perhaps obtain an estimate for the strength of the discrete vortices at the edges of the strip.

To avoid large gradients in the force density distribution at the edges of the strip, the force density is given by a function whose gradient at the edges can be adjusted. In the computations of the flow through the actuator strip a simple form of adaptive grid generation is used.

At each time step in the computation the grid points in the physical grid are redistributed according to the solution of the problem found in the preceding time step.

Two different configurations were calculated, the first being the half-infinite actuator strip. This is a strip with only one edge. The calculations with this configuration were used to test the convergence and the validity of the difference schemes that were used.

The second configuration, the finite actuator strip, is going to be used to determine the starting behaviour of the flow through the strip and to make an estimate of the value of the discrete vortices at the edges of the strip. This value is the yet missing link in the new theory presented in [ref.6].

The calculations are restricted to the situation with zero undisturbed velocity. This situation can be compared with the flow caused by a helicopter in hover.

The complexity of the programme to calculate the flow through the finite actuator strip made it not possible to complete the work within the normal time. Only some results of testing programmes can be presented in this report. However, the whole programme is complete now and is in the final state of debugging. The rest of the calculations will be presented in an addendum.

1. The Actuator Strip.

Solving the flow equations for flows around wings, propellers and bodies in general is very difficult because of the complexity of the flow.

Things could be made easier, if idealized models were used. Such a model for the flow through a rotor is the flow through an actuator disc.

The actuator disc is obtained from a rotor by taking limits: the number of blades is increased and the cord of the blades decreased in such a way,that their product is kept constant. The total area of the rotorblades that is "seen" by the flow, is kept constant.

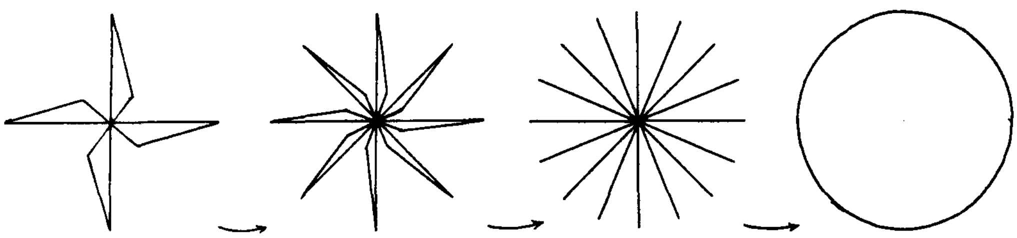


Fig.(1.1): Transition from real rotor to actuator disc.

The number of blades is infinite in the limit, the chord is zero. The force exerted on the air by the rotor (propellor) or vice versa (windturbine) is equally spread over the actuator strip. The force distribution over one blade divided by the radial distance to the axis equals the radial force distribution on the disc. This force is not necessarily constant along the radius of the disc. From now on we, however, assume a constant distribution of force density on the disc. The assumed force distribution on the blade then is proportional to the radial distance from the axis.

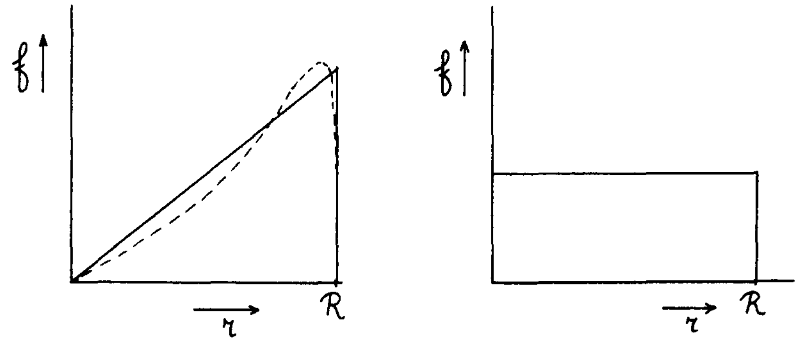
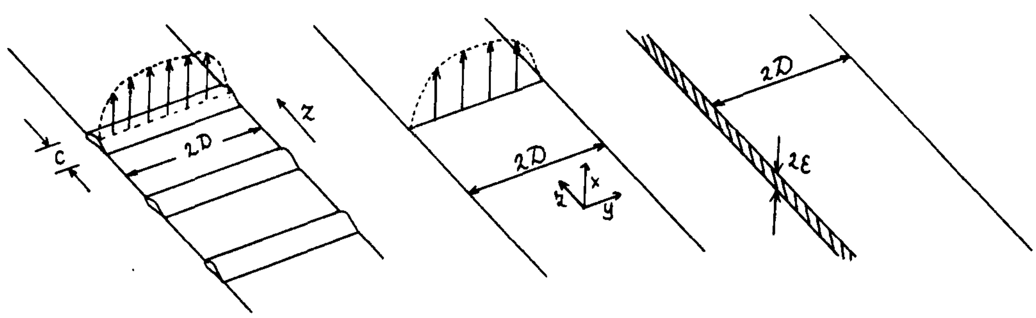


Fig.(1.2): Force distribution on rotor blade (left) in N/m and on the actuator disc in N/m^2. The dashed line gives the force distribution on a real blade.

Another way of obtaining the actuator strip is to place an infinite number of wing sections in succession in Z-direction, each with the same lift force distribution. By again increasing the number of wing sections while reducing the chord c, the actuator strip is born.



Figs.(1.3), (1.4a) and (1.4b): Placing infinitely many wingsections in a line while decreasing the chord c gives the actuator strip. The actuator strip that is used in the calculations actually has thickness 2ε.

Because the force density distribution is independent of the z-coordinate the flow around and through the actuator strip is a 2-dimensional one. This flow can be studied in an X-Y plane, with z = 0 for instance. If we again assume a constant force density distribution, we get a situation as depicted in figure 1.5. In the remainder we assume, that the force exerted by the strip, points only in the

±X-direction. The edges of the strip are at $y = \pm D$. The unit of force density is (N/m^3). The force density is also spread over a small layer with thickness 2ϵ, from $x = -\epsilon$ to $x = +\epsilon$.

Mathematically we can take the limit for $\epsilon \to 0$ and arrive at a force area 'density' in (N/m^2).

If we want to solve the problem numerically, however, the finite thickness of the layer has to be maintained, as shall be explained in chapter 3.

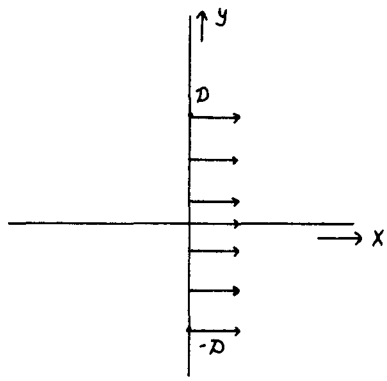


Fig.(1.5): Cross section of the actuator disc in the x-y plane. The force distribution is spread from $x = -\epsilon$ to $x = \epsilon$.

In X-direction a distribution of force density is chosen which is a well-known starting function to obtain the Dirac δ-function.

The function reads:

$$f(x,y) = f(y).f_o.\frac{n}{2\cosh^2(nx)} \quad (1.1)$$

The area under the function is given by:

$$\int_{-\infty}^{+\infty} f(x,y)\,dx = f(y).f_o.1/2.\tanh(nx)\Big|_{-\infty}^{+\infty} = f(y).f_o \quad (1.2)$$

By altering the parameter n one is able to vary the peakheight to width ratio without changing the area beneath the function. As we shall see later on, this is a useful possibility when calculating the flow numerically. Letting n go to infinite gives the Dirac δ-function.

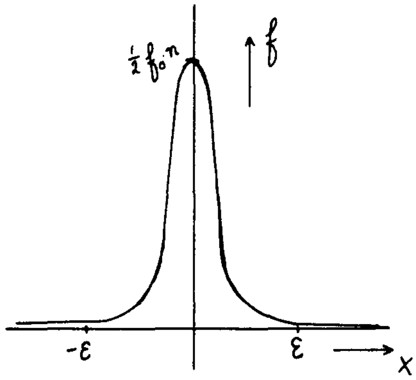


Fig.(1.6): Force density distribution as a function of x for y = 0.

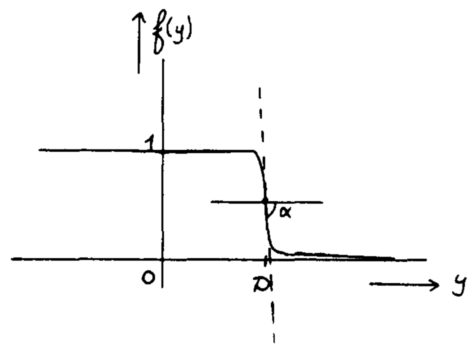


Fig.(1.7): Force density distribution as a function of y for x = 0.

At the edges, the force density falls graduately from maximum value to zero, but within a short distance. The Y-dependance at Y = + D is given by:

$$f(y) = 1/2.(\ 1-\tanh(m(y-D\ ))) \qquad (1.3)$$

Here m determines the distance over which the function falls from nearly maximum to nearly zero. The maximum gradient of the function is at y = D and equals $-1/2 \cdot m$.

For the edge at y = -D , the derivation of the function defining the force is similar.

2. The Flow Equations.

2.1 Equations.

A flow field in general is determined by the direction and absolute value of the velocity of elementary cells in the entire domain.

In other words: solving the flow equations comes down to determining the velocity field. One of the methods to determine the velocity field is to first solve the vorticity-diffusion equation, and then to calculate the streamfunction which gives the velocity field.

The flow through the actuator strip is caused by the action of the force density distribution in the strip.

The mathematical relationship between forces and velocity is given by the Navier-Stokes equation. This equation, assuming that air is incompressible and has only linear, isotropic shear, reads:

$$\frac{\partial \vec{u}}{\partial t} = \frac{1}{\rho_0}.\ \vec{f}(x,y) - (\vec{u}.\vec{\nabla}).\vec{u} + \nu.\vec{\nabla}^2.\vec{u} \qquad (2.1)$$

If we are interested in the production and transportation of vorticity, we take the rotation (the outer product with the differential operator $\vec{\nabla}$) of equation (2.1) and arrive at the Vorticity-Diffusion Equation (VDE); the Z-component of this equation reads:

$$\frac{\partial \omega_z}{\partial t} = \frac{1}{\rho_o}\left[-\frac{\partial f_x}{\partial y} + \frac{\partial f_y}{\partial x}\right] - (\vec{u}\cdot\vec{\nabla})\ \omega_z + \nu\ \vec{\nabla}^2\omega_z \qquad (2.2)$$

From now on, ω shall be used instead of ω_z.

The term on the left-hand side gives the change of the vorticity as a function of time, this change results from the terms on the right-hand side. The first one of these gives the production of vorticity by external forces, in our case by the gradient of volumetric force density at the edge(s) of the strip. The actuator strip is switched on at t = 0, this means that f(x,y) = 0 for $t \leq 0$. The second term gives the transportation of vorticity by the local velocity and is called the convective term. The latter term gives the transportation of vorticity by diffusion, the diffusion term.

The reader may ask himself, why equation (2.1) is not directly solved in order to get the velocity field. The reason for this is that we are also interested in the distribution of vorticity. As it turns out, the actuator strip produces vorticity at the edges. These vortices can be seen directly as peaks in the vorticity distribution. Working numerically with peaked functions is only possible, if the grid points are distributed in a special way. For this grid point distribution the vorticity distribution has to be known.

Back to the equations. Because the flow is incompressible and 2-dimensional, a streamfunction ψ can be defined with:

$$u = \frac{\partial \psi}{\partial y}, \qquad v = -\frac{\partial \psi}{\partial x} \qquad (2.3)$$

where u is the velocity in X-direction, v the velocity in Y-direction, and with the definition of ω, the vorticity in Z-direction:

$$\omega = (\vec{\nabla} \times \vec{u}) = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \qquad (2.4)$$

we find the relationship between ω and ψ, the so-called Poisson-PSI equation (PPE):

$$\nabla^2 \psi = - \omega \qquad (2.5)$$

By solving this equation for a given vorticity distribution, the accompanying streamfunction is found. With a known streamfunction, the velocity is found by applying the equations (2.3). Now the circle of equations, to be used in one time step, is closed.

In principle, a solution to equation (2.2) for a given time can be found. This could be done, for example, by an iterative process; an initial estimate of the velocity is used to solve the vorticity diffusion equation and the Poisson-PSI equation, which gives a new velocity distribution. This process is repeated untill the difference between initial and found velocity are small enough. Choosing this method would increase the time, neede to solve the time-dependent problem, very much.

So an additional assumption has to be made to override this difficulty. By only allowing small time increments Δt, the velocity is assumed to be quasi-stationary, so equation (2.2) can be solved at t_1 using the velocity at time t_0. This introduces an error in the computations, but this error can be reduced by decreasing the time step.

The equations (2.2) and (2.5) can only be solved, if some additional relationships for ω and ψ, respectively, are given at the boundaries of the domain. These relationships are called the boundary conditions. For the flow through a half-infinite actuator strip the far-field boundary condition has been used: vorticity and velocity are both zero at the boundaries. These boundaries are taken far away from the origin. In calculating the flow through the finite actuator strip the far field condition has been used for three boundaries. The condition for the fourth boundary was the mirror-condition: because the flow through the actuator strip is symmetrical only half the field has to be computed. At crossing the symmetry line the velocity in x-direction stays the same, while the velocity in y-direction and the vorticity change sign.

2.2 The differential schemes.

Because each domain has infinitely many points, it's impossible to solve any differential equation for each point numerically. For this reason, a grid is laid over the domain and the differential equation is transformed into a so-called differential scheme, a relationship between the function values of adjoining grid points. Let's take the function $\omega(x,y)$ as an example. The computational grid is regular (the distance between grid points is constant) and orthogonal (the lines linking grid points with the same X-index cross lines that link points with the same Y-index at right angles). There are m points in X- and n points in Y-direction. Each grid point is identified by an index l, with $l = (j-1)\cdot m + i$, where j is the number of the row and i the number of the column the grid point lies on. l ranges from 1 until $m\cdot n = L$. See fig. (2.2.1).

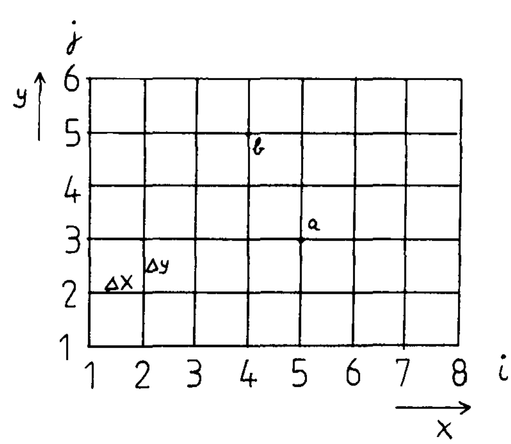


Fig.(2.2.1): Regular grid point distribution in a 8x6 points grid.m = 8, n = 6. In a l = 21, in b l = 36. l is given by : l = (j-1)•m + i.

If we consider a grid point l with its 8 direct adjacent points (fig.2.2.2), the vorticity in one of these points can be written as a Taylor series in the vorticity of the central point. For instance, if l is the index of the central point:

$$\omega_{l+1} = \omega_l + \Delta x \cdot \frac{\partial \omega}{\partial x}\bigg|_l + \frac{1}{2} \Delta x^2 \cdot \frac{\partial^2 \omega}{\partial x^2}\bigg|_l + \frac{1}{3!} \Delta x^3 \cdot \frac{\partial^3 \omega}{\partial x^3}\bigg|_l + O(\Delta x^4) \quad (2.6)$$

and

$$\omega_{l-1} = \omega_l - \Delta x \cdot \frac{\partial \omega}{\partial x}\bigg|_l + \frac{1}{2} \Delta x^2 \cdot \frac{\partial^2 \omega}{\partial x^2}\bigg|_l - \frac{1}{3!} \Delta x^3 \cdot \frac{\partial^3 \omega}{\partial x^3}\bigg|_l + O(\Delta x^4) \quad (2.7)$$

where $O(\Delta x^4)$ means, that the remainder of the series is of size Δx^4. This is the so-called truncation error of the Taylor series. If we add (2.6) and (2.7) we get:

$$\omega_{l+1} + \omega_{l-1} = 2 \cdot \omega_l + \Delta x^2 \cdot \frac{\partial^2 \omega}{\partial x^2}\bigg|_l + O(\Delta x^4) \quad (2.8)$$

Reordering of the terms gives:

$$\frac{\partial^2 \omega}{\partial x^2}\bigg|_l = \frac{\omega_{l+1} + \omega_{l-1} - 2 \cdot \omega_l}{\Delta x^2} - O(\Delta x^2) \quad (2.9)$$

Here we have the second derivative of the vorticity with respect to x, written in terms of the value of ω in the points l-1, l+1 and l itself.

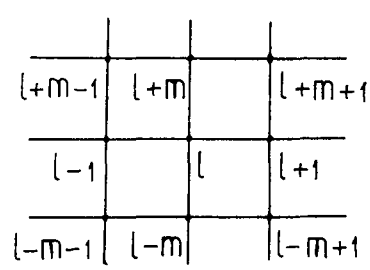


Fig.(2.2.2): Central grid point l with its 8 direct adjacent points.

Writing down Taylor series for the other grid points in the same manner and subtracting or adding them in such a way that certain terms vanish, gives us the discretisation schemes for a variety of derivatives:

$$\left.\frac{\partial\,\omega}{\partial\,x}\right|_{l} = \frac{\omega_{l+1} - \omega_{l-1}}{2\cdot\Delta x} + O(\Delta x^2) \qquad (2.10)$$

$$\left.\frac{\partial\,\omega}{\partial\,y}\right|_{l} = \frac{\omega_{l+m} - \omega_{l-m}}{2\cdot\Delta y} + O(\Delta y^2) \qquad (2.11)$$

$$\left.\frac{\partial^2\,\omega}{\partial\,y^2}\right|_{l} = \frac{\omega_{l+m} + \omega_{l-m} - 2\cdot\omega_{l}}{\Delta y^2} + O(\Delta y^2) \qquad (2.12)$$

$$\left.\frac{\partial^2\omega}{\partial x\partial y}\right|_{l} = \frac{\omega_{l-m-1} - \omega_{l-m+1} - \omega_{l+m-1} + \omega_{l+m+1}}{4\cdot\Delta x\cdot\Delta y} + O(\Delta x^2+\Delta y^2) \qquad (2.13)$$

If we now want to turn the Vorticity Diffusion Equation (VDE) into a differential scheme, all the derivatives in the equation have to be replaced by their discrete approximation, for instance by equations (2.9) up to and including (2.12), if a second order truncation error is satisfactory.

The time derivative in the VDE is replaced by an approximation with first order truncation error:

$$\left.\frac{\partial\,\omega}{\partial\,t}\right|_{l,t} = \frac{\omega_l^n - \omega_l^{n-1}}{\Delta t} + O(\Delta t) \qquad (2.14)$$

This is called the backward time difference. The upper index refers to the time: $t_n = t_0 + n\cdot\Delta t$, if no upper index is used, a time t_n (upper index $_n$) is assumed.

For the convective term in the VDE a special differential scheme is used in the programmes for the half infinite actuator strip. This scheme is described in section 3.2: 'The convective term in the differential scheme'.

The complete substitution in the VDE gives:

$$\frac{\omega_l - \omega_l^{n-1}}{\Delta t} = \frac{1}{\rho_o}\left[\frac{\partial f_y}{\partial x} - \frac{\partial f_x}{\partial y}\right] - u\cdot\frac{\omega_{l+1} - \omega_{l-1}}{2\cdot\Delta x} - v\cdot\frac{\omega_{l+m} - \omega_{l-m}}{2\cdot\Delta y} +$$

$$+ \nu\cdot\frac{\omega_{l+1} + \omega_{l-1} - 2\cdot\omega_l}{\Delta x^2} + \nu\cdot\frac{\omega_{l+m} + \omega_{l-m} - 2\cdot\omega_l}{\Delta y^2} + O(\Delta x^2+\Delta y^2+\Delta t) \quad (2.15)$$

Regrouping of the terms gives:

$$- \omega_l^{n-1} - \frac{\Delta t}{\rho_o}\cdot\left[\frac{\partial f_y}{\partial x} - \frac{\partial f_x}{\partial y}\right] = \Delta t\cdot\left[\frac{v}{2\cdot\Delta y} + \frac{\nu}{\Delta y^2}\right]\cdot\omega_{l-m} + \Delta t\cdot\left[\frac{u}{2\cdot\Delta x} + \frac{\nu}{\Delta x^2}\right]\cdot\omega_{l-1}$$

$$+ \left[\Delta t\cdot\left[\frac{-2\cdot\nu}{\Delta x^2} - \frac{2\cdot\nu}{\Delta y^2}\right] -1\right]\cdot\omega_l + \Delta t\cdot\left[\frac{-u}{2\cdot\Delta x} + \frac{\nu}{\Delta x^2}\right]\cdot\omega_{l+1} + \Delta t\cdot\left[\frac{-v}{2\cdot\Delta y} + \frac{\nu}{\Delta y^2}\right]\cdot\omega_{l+m}$$

(2.16)

With general coefficients the differential scheme for the Vorticity Diffusion Equation reads:

$$BO_l\cdot\omega_{l-m} + DO_l\cdot\omega_{l-1} + EO_l\cdot\omega_l + FO_l\cdot\omega_{l+1} + HO_l\cdot\omega_{l+m} = JO_l \quad (2.17)$$

This is a linear function that connects the value of ω in each grid point l with the value of ω in its four direct adjoining grid points. For this reason the scheme is called the five-point molecule scheme. The naming of the coefficients is systematically: the coefficients belonging to the value in the grid point l-m-1 begin with an A, the coefficients belonging to the value in grid point l-m with B etcetera. The second letter is either an O for coefficients of the vorticity diffusion equation (omega) or a P for the Poisson-PSI equation (psi).

The whole grid consists of L grid points, meaning that L equations have to be solved. If we write these set of L equations as one matrix equation, we obtain:

$$\bar{\bar{A}}\cdot\vec{\Omega} = \vec{J} \quad (2.18)$$

With $\bar{\bar{A}}$ a L x L matrix and $\vec{\Omega} = (\omega_1, \omega_2, \omega_3, \omega_4, \ldots, \omega_L)$ and $\vec{J} = (JO_1, JO_2, JO_3, JO_4, \ldots, JO_L)$.

The matrix $\bar{\bar{A}}$ is a so-called pentadiagonal matrix; this is a sparse matrix with only five diagonals nonzero: the main diagonal and its 2 direct adjacent diagonals, plus the two diagonals at a distance ± m from the main diagonal. (see fig.2.2.3) Each row l in the matrix is filled with the coefficients belonging to the discretized differential equation for gridpoint l.

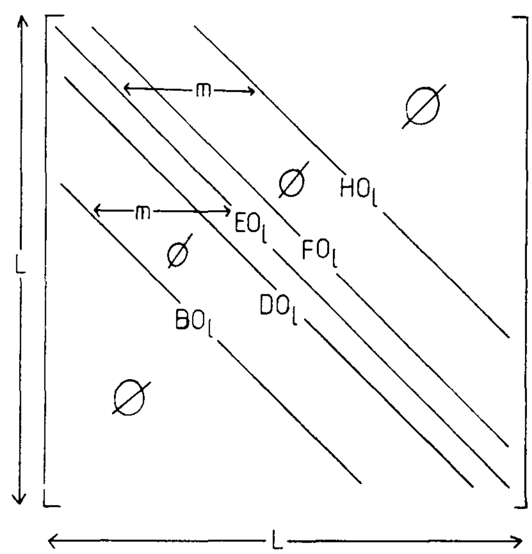


Fig.(2.2.3): Schematic representation of pentadiagonal matrix with L rows and L columns. Each row contains the coefficients of eq. 2.17 for the l-th gridpoint.

There are several numerical methods for solving these special kinds of matrix equations very quickly. The method used in this work is a method based on Stone's Strongly Implicit Method.(see [ref.3]).The flow through the half infinite strip is solved with the symmetric, strongly implicit method as described in [ref.4], and the flow through the finite actuator strip is solved using the modified strongly implicit method, described in [ref.5].

The replacement of the Poisson-PSI equation (PPE) by its differential scheme (also a five-point molecule scheme) is done in the same way. The result is:

$$- \psi_l = \frac{1}{\Delta y^2}\cdot\psi_{l-m} + \frac{1}{\Delta x^2}\cdot\psi_{l-1} + \left[\frac{-2}{\Delta x^2} - \frac{2}{\Delta y^2}\right]\cdot\psi_l + \frac{1}{\Delta x^2}\cdot\psi_{l+1} + \frac{1}{\Delta y^2}\cdot\psi_{l+m} \tag{2.19}$$

This set of equations is solved with the same method.

The velocity components, finally, are calculated with:

$$u_l = \frac{\psi_{l+m} - \psi_{l-m}}{2\cdot\Delta y} + O(\Delta y^2) \tag{2.20}$$

$$v_l = \frac{\psi_{l-1} - \psi_{l+1}}{2\cdot\Delta x} + O(\Delta x^2) \tag{2.21}$$

The differential schemes all have a truncation error of order $(\Delta x^2+\Delta y^2+\Delta t)$. This truncation error provides us with a means to check the correctness of the differential scheme itself. If, for instance, the time step is halved, keeping Δx and Δy constant, the truncation error also has to reduce to 50%. If we again reduce the time step with 50% and call the solutions to the equations with different time step a_r, a_{r+1} and a_{r+2} respectively, we can compare the differences in truncation error by subtracting the solutions.

If the errors are called ϵ, we have:

$$\epsilon_{r+2} - \epsilon_{r+1} = a_{r+2} - a_{r+1} \tag{2.22}$$

$$\epsilon_{r+1} - \epsilon_r = a_{r+1} - a_r \tag{2.23}$$

And if, as assumed $\epsilon_{r+1} = \frac{1}{2}\cdot\epsilon_r$, $\epsilon_{r+2} = \frac{1}{2}\cdot\epsilon_{r+1}$, then

$$\frac{\epsilon_{r+2} - \epsilon_{r+1}}{\epsilon_{r+1} - \epsilon_r} = \frac{1}{2} \tag{2.23a}$$

The convergence rate of an iterative process is defined as:

$$C_f = \frac{a_{r+2} - a_{r+1}}{a_{r+1} - a_r} \tag{2.24}$$

If $C_f = \frac{1}{2}$, then we can say that the method is of order zero with respect to time and the truncation error is of order 1.
The same procedure can be used to find the order of the solution method with respect to the spatial coordinates. The differential scheme suggests a truncation error of order 2 for both X- and Y-direction.

2.3 The boundary conditions.

As has already been mentioned, the partial differential equations can only be solved, if the behaviour of the function is given at the boundary. This can be understood quite easily, if we regard a grid point on the boundary (fig. 2.3.1). If some equation (e.g.(2.17)) for the point l is to be solved, the value of ω at l-1 has to be known.

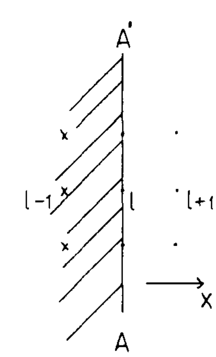


Fig.(2.3.1): Grid point l on boundary A-A'. The function value in points beyond the boundary can be expressed in function values of points inside by use of the boundary condition.

Normally, this value can be derived from values of points inside the domain by application of the boundary condition. For instance, if the line A-A' is a symmetry line the boundary condition would be $\frac{\partial \omega}{\partial x} = 0$ on the boundary.

In the computation of the flow through the finite actuator strip (ch.4), a similar boundary condition is used: there the X-axis is an anti-symmetry line and $\omega_{l-m} = -\omega_{l+m}$, if l is the index of a grid point that lies on the X-axis. The boundary conditions for the velocity are symmetrical: v = 0 and $\frac{\partial u}{\partial y} = 0$.

This leads to $\psi = 0$ on the X-axis.

The other boundary condition that has been used is the far-field condition: the vorticity as well as the velocity are zero far away from the actuator strip. This means $\omega_l = 0$, $u_l = 0$, $v_l = 0$, $\psi_l = 0$ if grid point l on the boundary, plus $\psi_l = 0$, if grid point l is immediately next to the boundary.

In case of nonzero undisturbed velocity $\vec{U} = U\cdot\vec{e}_x + V\cdot\vec{e}_y$ the boundary condition for the streamfunction becomes:

$$\psi = U\cdot y - V\cdot x \qquad \text{with } \psi(0,0) = 0 \tag{2.25}$$

The situation of nonzero undisturbed velocity can be compared with e.g. an ascending or descending helicopter or a windturbine.

3. The half-infinite actuator strip.

3.1 Model, source term and grid.

Model, source term.

If one edge of an actuator strip is held fixed on the origin and the other is moved to $y = -\infty$ while keeping the mean force density distribution constant, we get the half-infinite actuator strip. See fig.(3.1.1)

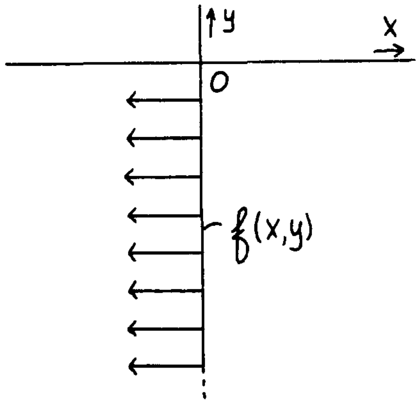


Fig.(3.1.1): Place of the edge and direction of the force for the half-infinite actuator strip

The first term on the right-hand side of equation (2.2) is the source term: we see that vorticity is only produced where a change of force density occurs. Because the half-infinite actuator strip only has one edge and the force has only an X-component, the only place where vorticity is produced, is at the origin. This production is given by:

$$\text{vorticity production} = -\frac{\partial f(x,y)_x}{\rho_o \cdot \partial y} \qquad (3.1)$$

The functional form of f(x,y) is already mentioned in chapter 1, formulae (1.1) and (1.3). Combined they give:

$$f(x,y) = -\frac{f_o \cdot n \cdot (1-\tanh(n \cdot y))}{\rho_o \cdot 4 \cdot \cosh^2(n \cdot x)} \qquad (3.2)$$

If the value of n is taken 4.6 , then at distances 2 from the centre of the source the force density has diminished to less than 1/1000 th. of the maximum value.

If we for a moment only consider the production of vorticity by the source and leave out the influence of diffusion and convection, the produced vorticity field will have the shape of a peak in the origin. This peak is an approximation of the vorticity field of a point vortex. So the corresponding velocity field is approximately the field of a point vortex in the origin, especially in regions further away from the origin. Because the streamlines of a point vortex are concentric circles there will be no transportation of vorticity in radial direction by convection. This is true only as a result of diffusion.

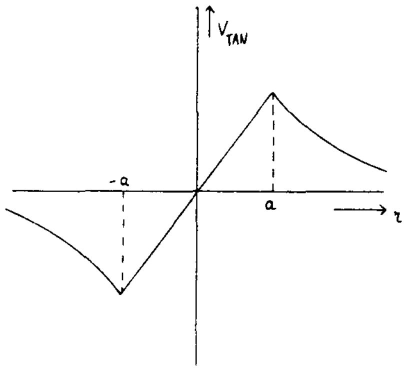


Fig.(3.1.2): Velocity field of a vortex with 'solid core'.

Another functional form of the source term is proposed by Gijs van Kuik [ref.6] and reads:

$$f = f_o \cdot \frac{1}{2} \cdot \left[1 - \frac{y}{|y|} \cdot \frac{y^2}{a^2} \right] \qquad \text{for } -a < y < a \qquad (3.3)$$

$$f = f_o \qquad \text{for } \quad y \leq -a \qquad (3.4)$$

$$f = 0 \qquad \text{for } \quad y \geq a \qquad (3.5)$$

This form is chosen because application in the inviscid flow equations gives a velocity field that is identical to the velocity field of a vortex with a 'solid core'. Only the tangential velocity is nonzero and is given in fig.(3.1.2). In the core with radius a the velocity is proportional to the distance from the origin, this is the velocity pattern of a rotating, massive body. Outside the core the velocity decreases proportional to 1/r, r being the distance to the origin. This is the velocity field of a point vortex in the origin. In the calculations a value for a of 1.5625 is taken, this is the distance between the third gridpoint and the origin. The source term for the source with solid core now reads:

$$\frac{\partial f}{\partial y} = - \frac{2.3}{\cosh^2(4.6 \cdot x)} \cdot \frac{y}{|y|} \cdot \frac{y^2}{a^2} \qquad (3.6)$$

this formula from now on will be referred to as the solid core source term.

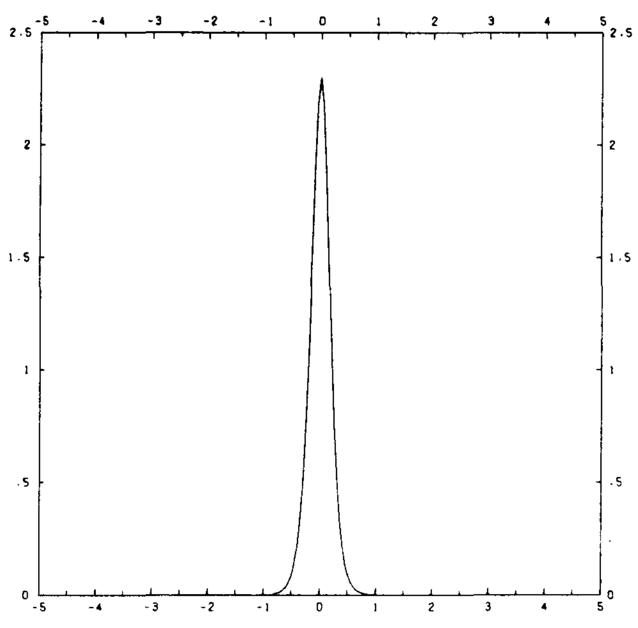


Fig.(3.1.3): Force density distribution in y-direction for cosh² source term.

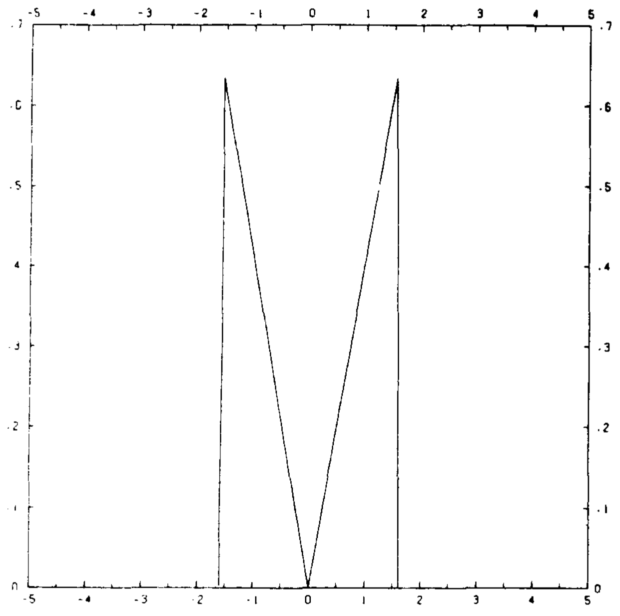


Fig.(3.1.4): Force density distribution in y-direction for solid core source term.

In the X-direction, a $\cosh^2$-dependence of the force density is introduced, as in the $\cosh^2$ source term. The figures (3.1.3) and (3.1.4) give the force density as a function of y for the $\cosh^2$-source and the solid core source respectively. Notice that the area beneath the function in both cases is 1.

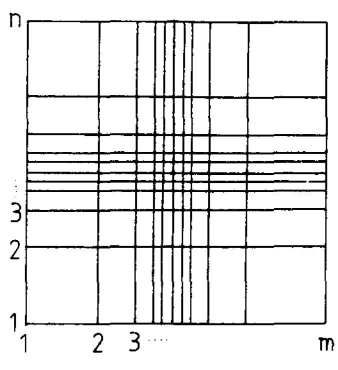


Fig.(3.1.5): Grid point distribution for computation of the flow through the half infinite actuator strip.

Grid.

The grid points in the computational grid are arranged in a special way. The grid consists of straight lines connecting points with same X- or Y-index. The distance between grid points decreases when the gridpoints approach either X- or Y-axis (see fig.3.1.1). The x- and y-values range between -100 and +100. Each row of gridpoints has y-coordinate y_j, j between 1 and n, and each column has x-coordinate x_i, i between 1 and m. x_i is given by:

$$x_i = 100 \cdot \left[\frac{i-(m+1)/2}{(m-1)/2}\right]^{(2 \cdot NVF+1)} \qquad (3.7)$$

where NVF is a positive integer.

The same relation holds for y_j:

$$y_j = 100 \cdot \left[\frac{j-(n+1)/2}{(n-1)/2}\right]^{(2 \cdot NVF+1)} \qquad (3.8)$$

When NVF = 1 we already have a strong decease in distance between gridpoints when approaching the axes.(square proportionality).For a 33 x 33 point grid, as was used in the computations, the x- (and y-) values for the consequent gridpoints are given in table 3.1.

i,(j)	x_i (y_j)
1 (33)	-100
2 (32)	-82.998
3 (31)	-66.992
4 (30)	-53.638
5 (29)	-42.188
6 (28)	-32.495
7 (27)	-24.414
8 (26)	-17.798
9 (25)	-12.5
10 (24)	-8.374
11 (23)	-5.273
12 (22)	-3.052
13 (21)	-1.563
14 (20)	-0.659
15 (19)	-0.195
16 (18)	-0.024
17	0

Table 3.1: Values of x_i and y_j for the gridpoints in a 33 x 33 grid with NVF=1. For the points 18 until 33 the values of the coordinates are positive.

3.2 The convective term in the differential scheme.

For calculation of the flow through the half-infinite actuator strip, the differential schemes of the flow equation, as described in section 2.2, are used, with one exception though. The convective term in the vorticity-diffusion equation has been discretised according to Roache [ref.1,p.73] in the so-called second upwind differencing scheme. This method assures conservation of vorticity as well as conservation of direction of convection.

The convective term is assumed to be quasi-stationairy. The vorticity in one grid point is developed as a series in the vorticity of neighbouring points in a way depending on the direction of the velocities in these points. The convective term can be written as:

$$(\vec{u}\cdot\vec{\nabla})\cdot\omega = u\cdot\frac{\partial\,\omega}{\partial\,x} + v\cdot\frac{\partial\,\omega}{\partial\,y} \tag{3.9}$$

If we take the first term on the right-hand side and write it according to the second upwind differencing scheme:

$$u\cdot\frac{\partial\,\omega}{\partial\,x} = \frac{u_r\omega_r - u_l\omega_l}{KX} \tag{3.10}$$

with

$$u_r = \frac{1}{2}\cdot\left[u_{i+1,j} - u_{i,j}\right] \tag{3.11}$$

$$u_l = \frac{1}{2}\cdot\left[u_{i,j} - u_{i-1,j}\right] \tag{3.12}$$

see fig.(3.2.1).

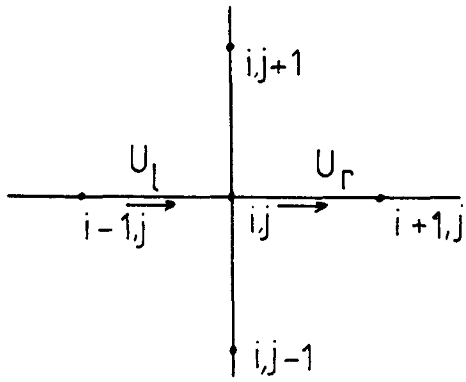


Fig.(3.2.1): definition of U_L and U_R

The values of KX, ω_r and ω_l are chosen depending on u_l and u_r :

$u_r \geq 0$ and $u_l \geq 0$: $KX = x_i - x_{i-1}$

$u_r < 0$ and $u_l < 0$: $KX = x_{i+1} - x_i$

otherwise: $KX = \frac{1}{2}\cdot(x_{i+1} - x_{i-1})$

and:

$u_r \geq 0 : \omega_r = \omega_{i,j}$ $\quad$ $u_r < 0 : \omega_r = \omega_{i+1,j}$

$u_l \geq 0 : \omega_l = \omega_{i-1,j}$ $\quad$ $u_l < 0 : \omega_l = \omega_{i,j}$

The scheme for $v\cdot\frac{\partial \omega}{\partial y}$ is the same, with i and j interchanged and x replaced by y.

3.3 Calculations and results.

On the half infinite actuator strip several calculations have been made. The bulk of these calculations was used to determine the convergence of the differential scheme. This is described in the next section.

For the strips with $\cosh^2$-source term and solid-core source term (see section 3.1) calculations have been made in a 33 × 33 point grid, with NVF = 1. (see section 3.1).

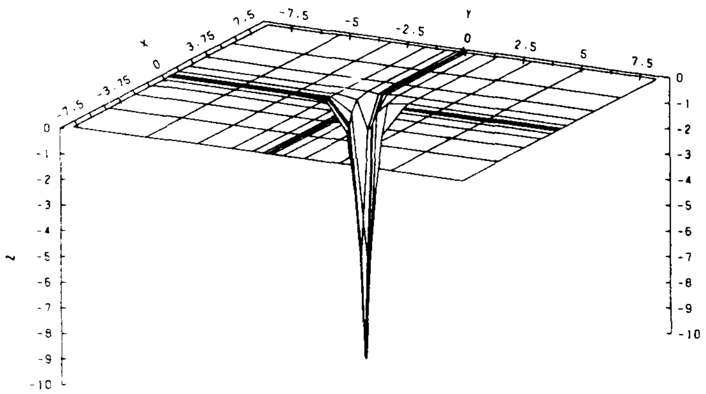


Fig.(3.3.1): Vorticity field at t=1.5 s. for the cosh² source term. Notice: vorticity is negative

Let's first turn to the strip with cosh²-source term.

Figure (3.3.1) gives a perspective view of the vorticity field that was calculated at t = 1.5 seconds in 10 time steps. The strength of the source was 100; this means that f_o in formula (3.2) is 100. Notice that the grid is not completely shown, both x and y extend from −100 to +100. The height of the peak is normalized to 10; the real value of the vorticity at the peak is ≃ 40. Fig.(3.3.2) shows the corresponding streamline pattern. As expected, the streamlines clearly form concentric circles.

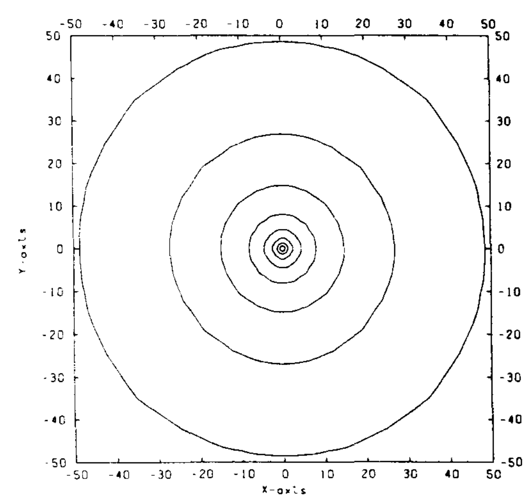
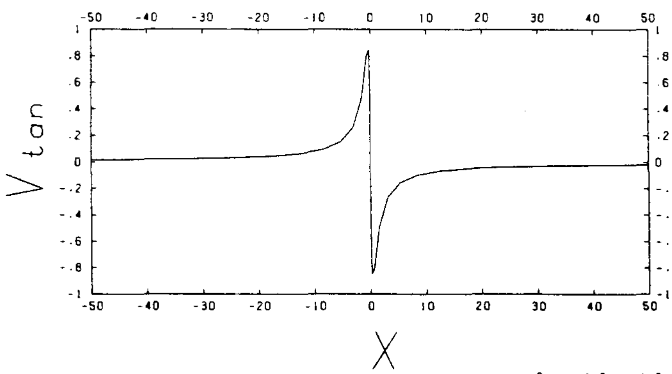
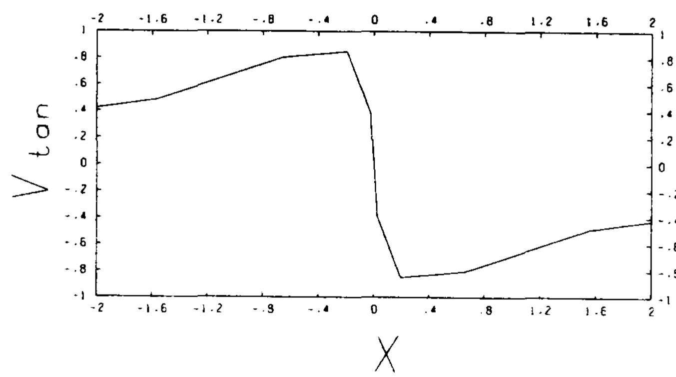


Fig.(3.3.2): Streamline pattern at t = 1.5 s. for cosh² source.

To find out, whether the tangential velocity has a reciprocal dependence of radial distance to the source, as is the case if the source is a real point vortex, we view figs. (3.3.3) and (3.3.4), which show the tangential velocity on the X-axis.





Figs. (3.3.3) and (3.3.4): Tangential velocity at t = 1.5 seconds as a function of x for the $\cosh^2$ *source.*

The same results for the strip with a solid core source term are shown in figs. (3.3.5) up to and including (3.3.8). Here the calculations are performed until t = 1.5 seconds, in again 10 time steps and with the same strength of the source. Notice the double-peaked vorticity distribution in fig.(3.3.5).

Fig.(3.3.5): *Vorticity field at t = 1.5 s for the solid core source term.*

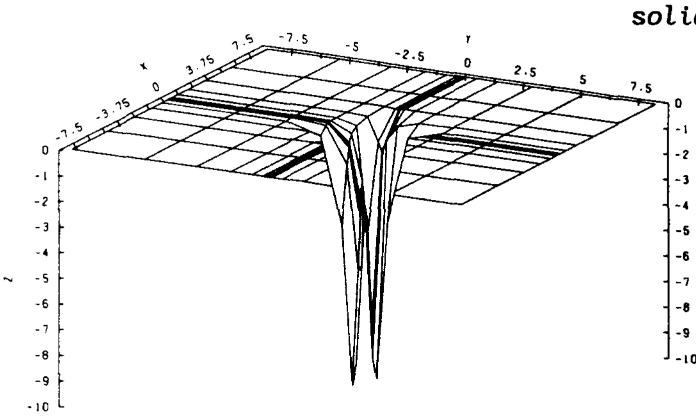
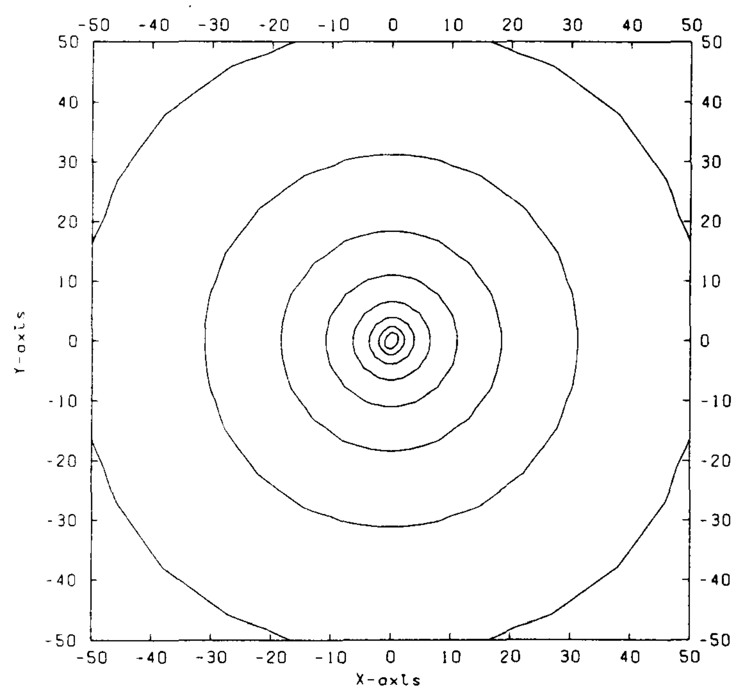
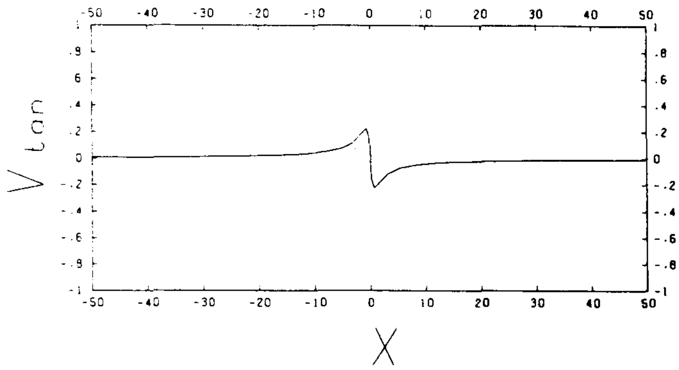
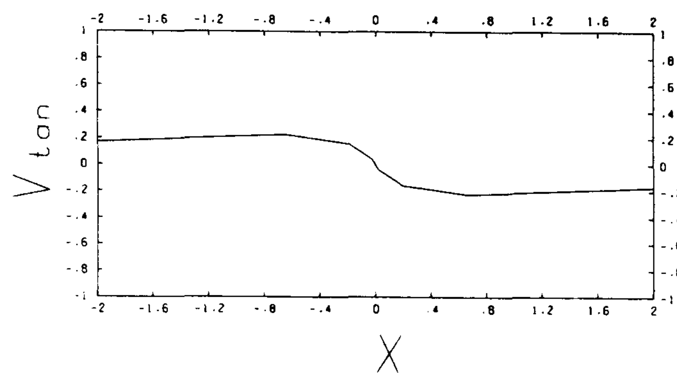


Fig.(3.3.6): *Streamline pattern at t = 1.5 s. for the solid core source term*







Figs.(3.3.7) and (3.3.8): Tangential velocity at t = 1.5 s. for the solid core source.

Because the dependency of v_{tan} from r, the radial distance to the source, is rather difficult to derive from fig.(3.3.3) or (3.3.7), figure (3.3.9) gives the product of v_{tan} and r as a function of r for both types of source. We see that in a large area the product hardly changes and that the departure from a straight line at small r is earlier for the solid core source than for the $\cosh^2$ source.

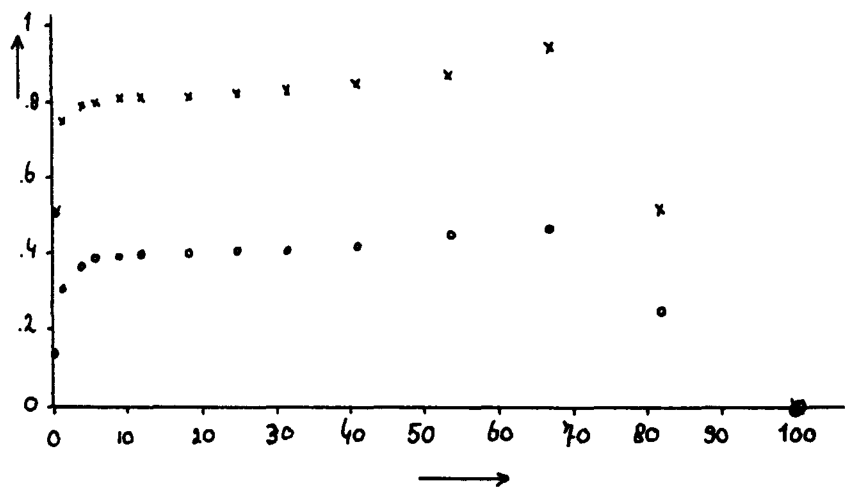


Fig.(3.3.9): $V_{tan} \cdot r$ as a function of r for two different source terms in the half infinite actuator strip. crosses: solid core; dots: $cosh^2$.

For large r the line is slowly inclining, but above $r \approx 70$ the curve breaks down. This downbreak for large r can be explained by the boundary condition for the velocity. These are both set to zero at the boundaries, so the velocity has to decrease at some point more quickly than the increase in r.

Here we arrive also at a possible explanation for the slight rise in the product $v_{tan} \cdot r$ for intermediate r. Because the vorticity is always set zero at the boundaries, there is no possibility for vorticity to escape the domain. Consequently, the velocity will increase more than it would, if vorticity could diffuse or convect out of the domain.

This effect could be avoided, if either a large domain were chosen, or if a smaller ending time of the calculations were chosen.

If we look at the course of $v_{tan} \cdot r$ for small r in fig.(3.3.9), we see that the departure from constant $v_{tan} \cdot r$ happens at larger r for the solid core source. This indicates that, although the same grid is used for both types of source, the solid core source has a larger core (i.e. a region where $v_{tan} \cdot r$ is not constant) than the $\cosh^2$-core. Probably, the size of the latter is dictated by the size of the mesh near the origin.

0.100	-1.433	-14.330
0.200	-2.866	-14.329
0.300	-4.299	-14.329
0.400	-5.731	-14.328
0.500	-7.164	-14.328
0.600	-8.597	-14.328
0.700	-10.029	-14.327
0.800	-11.461	-14.327
0.900	-12.894	-14.326
1.000	-14.326	-14.326
1.100	-15.757	-14.325
1.200	-17.189	-14.324
1.300	-18.620	-14.323
1.400	-20.050	-14.321
1.500	-21.478	-14.319
1.600	-22.904	-14.315
1.700	-24.328	-14.310
1.800	-25.744	-14.302
1.900	-27.150	-14.290
2.000	-28.551	-14.275

Table 3.2: Value of the vorticity as a function of time. First column: time; second column: vorticity; third column: vorticity over time.

As a final result, table 3.2 is given. Here the vorticity in the origin as afunction of time is given. The calculations were made in a 11x11 point grid. As can be read in the third column, the vorticity increases linearly as a function of time.

3.4 Convergence.

The calculations on the half-infinite actuator strip have also been performed to gain some insight into the convergence of the iteration process. According to the differential scheme, the order of the method to be expected is 1 for the spatial coordinates and 0 for the time coordinate. This order can be calculated by comparing the results of computations with consequently diminishing steps in x- or y- or t-direction.

For instance, if we want to know the order of the truncation error of the method for the x-coordinate, we first solve the problem in a 9 × 9 point grid and in, e.g., 2 time steps. Then we again solve the problem with the same parameters, except for the number of steps in x-direction, which is doubled. The grid is then 17 × 9 points. Then the problem is solved for the third time, now in a 33 × 9 point grid.

For each point in the original 9 × 9 point grid the convergence rate C_f can be calculated by applying equation (2.24). The order is minus the second logarithm of the convergence rate. For order 1 and 2 of the truncation error this factor has to be 1/2 and 1/4, respectively.

The convergence rate in a grid point was only calculated if the difference between consequent solutions was larger than the accuracy in the calculation.

These accuracies are EPSOM and EPSPSI in the calculation of the vorticity diffusion equation and the Poisson-PSI equation, respectively.

In the following three tables, the mean value of the calculated convergence rates, the standard deviations and the number of points in which the rates could be determined are printed.

For the determination of a set of convergence rates three calculations had to be performed, each next calculation with a double number of grid points in one direction.

An asterisk means, that the convergence rate in at least one grid point was negative.

At the top of each column the number of grid points that was used to determine the convergence rates is printed.

x-direction (17 points in y-direction, 2 time steps)

		9/17/33	17/33/65	33/65/129	Cf expected
ω	$\overline{Cf}$	9.2	0.45*	0.17*	0.25
	σ(Cf)	9.2	1.43	0.42	
	n	5	5	3	
u	$\overline{Cf}$	0.06*	1.39	0.33	0.25
	σ(Cf)	0.08	1.86	0.04	
	n	44	44	40	
v	$\overline{Cf}$	0.23*	0.75	0.29	0.25
	σ(Cf)	0.26	1.03	0.3	
	n	43	44	41	

$\varepsilon_\omega = 10^{-8}$
$\varepsilon_\psi = 10^{-5}$

Table 3.3: Convergence rates in x-direction.

y-direction (33 points in x-direction, 2 time steps)

		9/17/33	17/33/65	Cf expected
ω	$\overline{Cf}$	0.134	4.454	0.25
	σ(Cf)	0.16	-	
	n	3	1	
u	$\overline{Cf}$	0.27	0.52	0.25
	σ(Cf)	0.30	0.18	
	n	38	30	
v	$\overline{Cf}$	0.11	0.62	0.25
	σ(Cf)	0.01	0.13	
	n	30	26	

$\varepsilon_\omega = 10^{-5}$
$\varepsilon_\psi = 10^{-4}$

Table 3.4: Convergence rates in y-direction.

time (17x17 grid points)

		4/8/16	8/16/32	16/32/64	Cf expected
ω	$\overline{Cf}$	0.87	0.02	0.68	0.5
	$\sigma(Cf)$	0.57	0.8	0.18	
	n	13	13	9	
u	$\overline{Cf}$	0.376	0.420	0.455	0.5
	$\sigma(Cf)$	0.003	0.005	0.005	
	n	44	44	44	
v	$\overline{Cf}$	0.376	0.420	0.454	0.5
	$\sigma(Cf)$	0.003	0.005	0.005	
	n	44	44	44	

$\varepsilon_\omega = 10^{-10}$
$\varepsilon_\psi = 10^{-6}$

Table 3.5: Convergence rates in t-direction.

For diminishing step in x-direction, as well as in y-direction, the convergence rates in the calculation of the vorticity are not as good as could be expected from the order of the method. Probably this is due to the small number of grid points included in the calculation of the convergence rate.

The convergence rates for the velocities approach the expected values.

In case of diminishing time step, the convergence rate for the vorticity are better, and the rates for the velocities are even very good.

3.5 Comparison with an analytical solution.

If the vorticity diffusion equation is written in cylindrical coordinates (r,ϑ,z) without convection term, it reads:

$$\frac{\partial \omega}{\partial t} = - \nu \frac{1}{r}\frac{\partial}{\partial r}\left(r \frac{\partial \omega}{\partial r}\right) = f(r,\vartheta,t) \qquad (3.5.1)$$

If $f(r,\vartheta,t)$ is a unit point source in time and in space ($r=0$, $t=0$), the solution to (3.5.1) is:

$$\omega(r,\vartheta,t) = \frac{1}{\nu\, t}\cdot\exp(-r^2/4\nu t) \qquad (3.5.2)$$

(see [ref.15],pp.258-)

If the δ-function form is maintained in spatial coordinates, but the source is turned on from $t = 0$ until $t = t'$, then the source term reads:

$$f(r,\vartheta,t) = \delta_r(0)H(0,t') \qquad (3.5.3)$$

where $H(0,t')=1$ for $0\leq t\leq t'$ and $H(0,t')=0$ otherwise.

The solution to the equation (3.5.1) then is:

$$\omega(r,\vartheta,t) = \int_0^{t'} \frac{\exp(-r^2/4\nu t)}{\nu t}dt = \frac{1}{\nu}\int_{-r^2/4\nu t'}^{\infty} \frac{\exp(-p)}{p}dp = \frac{1}{\nu}\cdot E_1(r^2/4\nu t')$$

(3.5.4)

E_1 is the exponential integral. This function is tabulated in e.g. [ref.16], pp.228-.

For $p = 10$, $E_1(p) = 4.1\times10^{-6}$. This means. that for $t'= 1$, as in the numerical calculations, the vorticity at places with $r^2/4\nu = 10$ has a value of about 2.8×10^{-1}. This is for $r = 2.4\times10^{-2}$.

In a 33x33 point grid the smallest distance is approximately 2×10^{-2}. Here the calculated vorticity is -5.0. This is much larger than the value caused by a pure delta-function in the origin. The difference between these two values can be explained partly by the absence of diffusion in the analytical solution, and partly by the fact that the source term used in the calculations is an approximation of the Dirac δ-function. Therefore the gradients in the solution can be expected to be much smaller.

If $f(r,\vartheta,t)$ is given by:

$$f(r,\vartheta,t) = \frac{n}{2\cosh^2(nr')}H(0,t') \qquad (3.5.5)$$

the solution will be given by:

$$\omega(r,\vartheta,t) = \int_0^{\infty}\int_0^{t'}\int_0^{2\pi} \frac{n\cdot\exp(-r^2-r'^2+2rr'\cos(\vartheta-\vartheta')/4\nu t')}{2\cdot\cosh^2(nr')} d\vartheta' dt' dr' \qquad (3.5.6)$$

This solution isn't worked out further.

4. The (finite) actuator strip; Grid Generation.

4.1 Model of the flow through the actuator strip.

The finite actuator strip has two edges: at $x=0$, $y= \pm D$. In the ideal case, the force density at the edges falls from maximum to zero in an infinitesimal distance. The vorticity distribution formed in this case would consist of two infinitely high peaks at the edges.

In this model I use an approximation for the step in force density at the edges, given by eq.(1.3).

The vorticity will be formed in a small region around the edges. The area of this region depends on the parameter m in eq.(1.3): for increasing m, the area decreases and the maximum of the produced vorticity distribution increases.

The force exerted by the actuator strip will be assumed to point in positive x-direction and is turned on at $t = 0$. Then, according to equation (2.2), the vorticity produced at $y = + D$ is positive and negative at $y = -D$.

The initial vorticity distribution has an accompanying velocity distribution: the peak of vorticity at one edge will create a velocity in the + x-direction at the opposite edge and vice versa. Thus the vorticity produced at the edges of the strip will initially be transported in +x-direction.

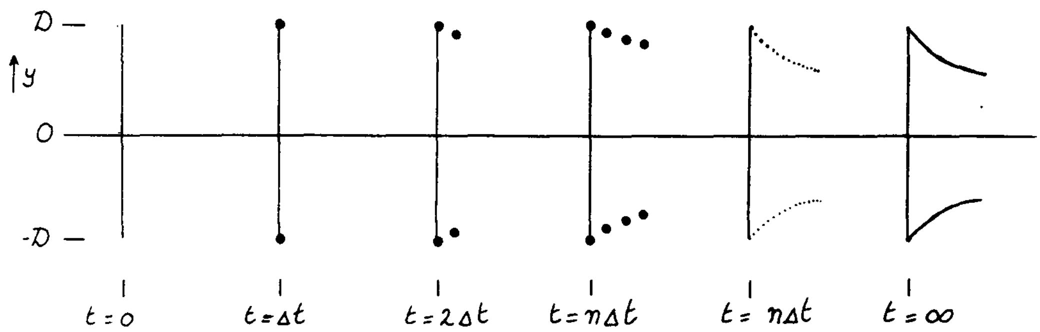


Fig.(4.1.1): Process of formation of vortex sheets at the edges of the actuator strip

This proces can also be made clear as follows. If we for a moment think that the vorticity that is produced at one edge in one time step, is concentrated in a single point vortex, it will be moved in +x-direction by the velocity field of the point vortex at the opposite edge. In the following time step, two new point vortices will be created at the edges and each vortex will be transported by the sum of velocities of the other three vortices. After a while, a whole chain of point vortices is formed to the right of each edge. (see fig.(4.1.1.)).If we let the time step decrease to zero, the distance between the individual point vortices will also become zero and the vortices will form a vortex sheet. This vortex sheet has zero thickness and infinite vorticity. In the model the vorticity is nowhere infinite and the vortex sheet has finite thickness, and is represented by a line of local maxima in the vorticity distribution.

After some time, a steady state situation will occur and the vortex sheets formed will have a constant shape. In this situation, the velocity at the edge of the strip is not unambiguously pointing in x-direction; the vortex sheet has a spiraling form in the vicinity of the edge.

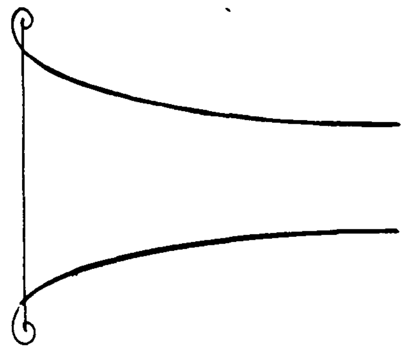
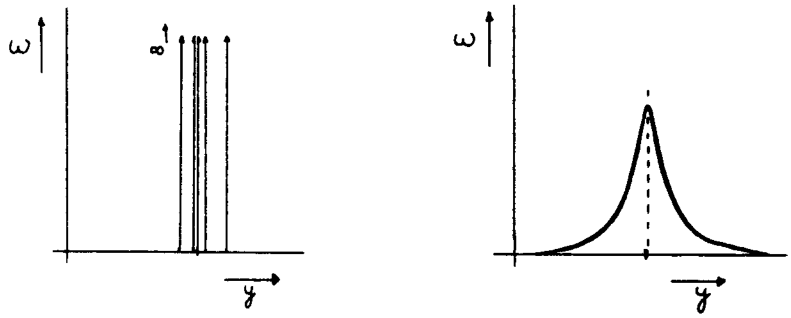


Fig.(4.1.2): Shape of vortex sheets in steady state.

The flow through an actuator strip without diffusion in the stationary state can also be calculated analytically. The problem then is transformed into a complex plane. For an actuator strip with constant force density distribution Schmidt & Sparenberg [ref.10] find as a solution a vortex sheet that spirals into the edge of the actuator strip (see fig.4.1.2). This solution is discussed by Gijs van Kuik [ref.6] who also finds a spiraling solution but with another behaviour in the limit of going to the edge.In these works, further references can be found to other works on this subject.

With the underlying model one is not able to obtain such a spiraling solution at the edges because of the minimum mesh size. The vorticity in a grid point will allways be some average value of the analytical solution. The vorticity distribution for $x = 0$, $t = \infty$ as a function of y is sketched in figs.(4.1.3) and (4.1.4). The first gives the analytical solution, the latter the average of the first. Figure (4.1.3) is a cross-section through the different branches of the vortex sheet, that spirals into the edge. The solution in (4.1.4) will be more accurate, if more grid points are used in those regions where the vorticity changes most.



Figs.(4.1.3) and (4.1.4): Vorticity distribution as a function of y for x = 0. left: analytical solution; infinitely high peaks; right: mean value of analytical solution.

Because we are interested in the starting behaviour of the flow through the actuator strip, we have to use the time-dependent equation (2.2). In this non-stationary case the vorticity produced at the edges is transported by the induced velocity field, so the vorticity distribution changes every time step.

If the flow is to be computed with maximum accuracy, the grid points have to be redistributed every time step in order to keep track of the moving vorticity distribution. So we would like to have a method that makes a grid of which the gridpoints are distributed dependent on the vorticity distribution. This method is called adaptive grid generation and is described in general in the next section.

4.2 Numerical Grid Generation.

If a problem is to be solved numerically in a domain with complicated boundaries, numerical grid generation can be used to transform the domain into another domain with simple boundaries. For instance, in fig.(4.2.1) a domain with curved boundaries is transformed into a rectangular domain.

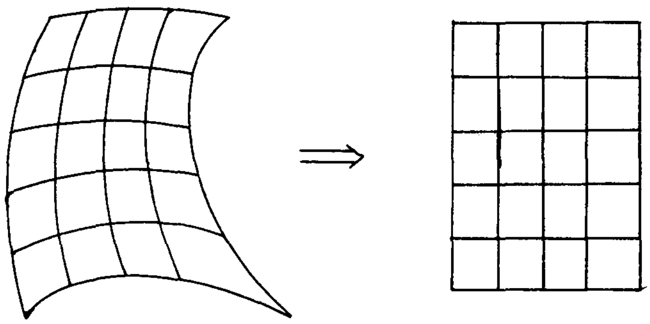


Fig.(4.2.1): Grid generation used to transform curved boundaries into straight ones.

Another possibility to use grid generation is in those situations where the solution is strongly varying throughout the domain. Grid generation then can be used to concentrate the gridpoints at places where for instance the solution has a large gradient (see fig.(4.2.2)).

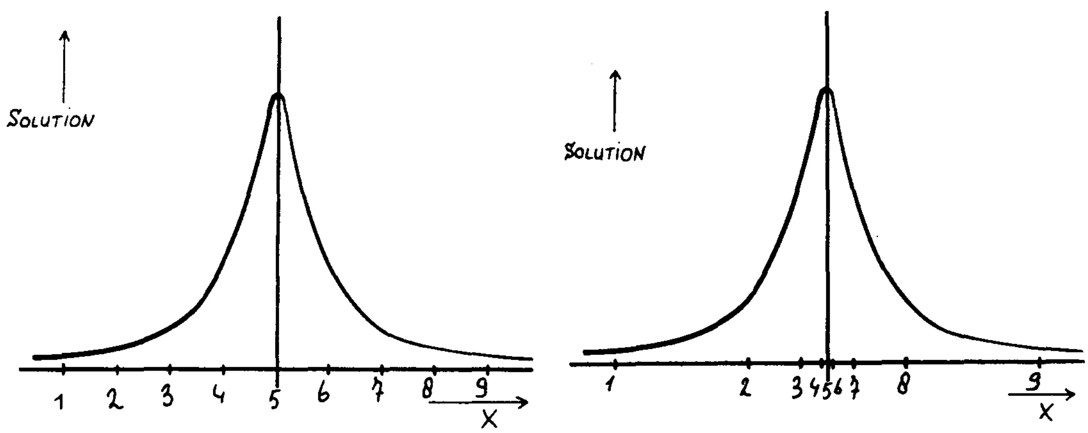


Fig.(4.2.2): Grid generation used to redistribute grid points on x-axis according to solution.

In general, grid generation leads to a quicker solution of the problem. In some cases grid generation offers a solution, and working with a regular grid doesn't.

Some general information about grid generation is given in [ref.7] and [ref.8]. However, the approach used in this work is much more simple and straightforward.

In the underlying problem, grid generation is used to redistribute the grid points over the domain. This redistribution is done every time step dependent on the solution obtained in the previous step and is therefore called adaptive grid generation.

Grid generation implies a transformation from a grid with grid points distributed in a special way (the physical grid) into a regular, orthogonal grid (the computational grid). The coordinates of the transformed grid (ξ,η) are written as a function of the coordinates in the physical grid (x,y):

$$\left.\begin{aligned} \xi &= \xi(x,y) \\ \eta &= \eta(x,y) \end{aligned}\right| \qquad (4.2.1)$$

These two functions define the transformation.

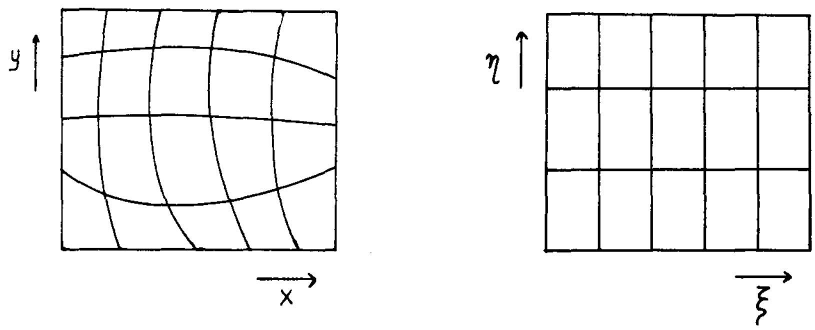


Fig.(4.2.3): Transformation of physical grid (left) into computational grid.

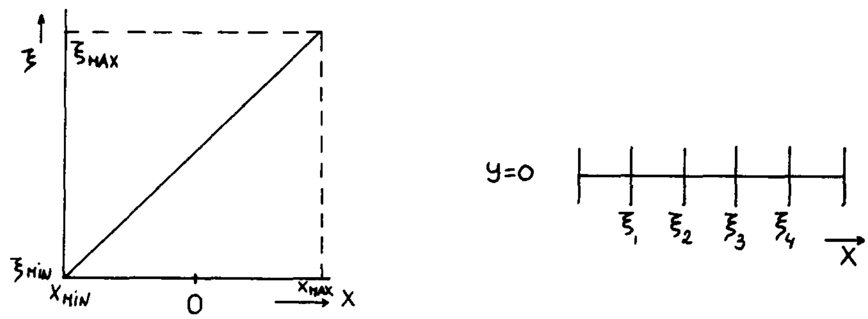
In figure (4.2.3) at the left, the physical grid is drawn, with drawn in it some lines ξ = constant and η = constant. These lines are transformed into straight lines in the computational grid. The difficulty is to construct functions ξ and η properly.

In analytical grid generation, ξ and η are written explicitly as a combination of analytical functions of x and y, for instance:

$$\xi = x^{1/n} \quad , \quad \eta = y \qquad (4.2.2)$$

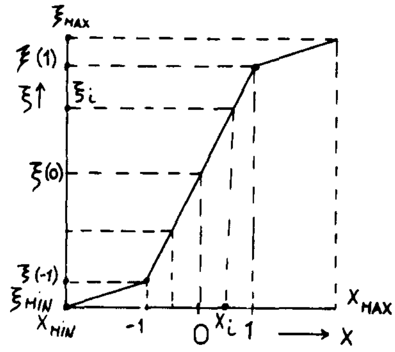
Because ξ and η are explicit functions of x and y, it's possible to find the inverse transformations $x = x(\xi,\eta)$ and $y = y(\xi,\eta)$ in a general way. If then the computational grid is set up, the physical grid can be calculated by applying the inverse transformation.

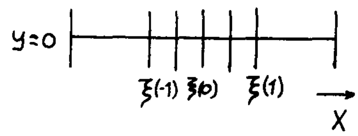
Another way of generating a grid is numerical grid generation. Now only relationships between individual points in physical and computational grid are given. The inverse transformation can be found by interpolation or by fitting a general function (e.g. a power series) to the set of points.



Figs.(4.2.4) and (4.2.5): Relation between ξ and x linear. ξ is equally distributed on the x-axis.

Let's regard the relationship between ξ and x for fixed y (e.g. $y = 0$). For ξ and x both regularly distributed over the domain, the transformation is linear and can be depicted as a straight line in the x-ξ plane.(fig.(4.2.4)). If $\xi_{min} = x_{min}$ and $\xi_{max} = x_{max}$, the partial derivative of ξ with respect to x is 1 everywhere, meaning that the lines ξ = constant are equally distributed in the physical grid for $y = 0$. (fig.4.2.5).





Figs. (4.2.6) and (4.2.7): Contraction of grid points between -1 and 1. Here the partial derivative of ξ with respect to x is larger than in the rest of the domain.

If we want to have more grid points in the physical grid between, for example, x = -1 and x = 1, we can choose the partial derivative of ξ with respect to x larger in this interval. This also implies a smaller ξ_x in the rest of the domain because:

$$\int_{x_{min}}^{x_{max}} \xi_x \, dx = \xi_{max} - \xi_{min} = \text{constant} \qquad (4.2.3)$$

If only the values of ξ and x are known in the points x = x_{min}, -1, 1, and x_{max}, the x-values of the other points in the computational grid can be found by interpolating between the known values.

For instance, linear interpolation in fig.(4.2.6) gives for x_i:

$$x_i = -1 + \frac{\xi_i - \xi(-1)}{\xi(1) - \xi(-1)} \cdot (1 - (-1)) \qquad (4.2.4)$$

In general, if i is the index for the subsequent points on the x- or ξ-axis:

$$x' = x_{i-1} + \frac{\xi' - \xi_{i-1}}{\xi_i - \xi_{i-1}} \cdot (x_i - x_{i-1}) \qquad (4.2.5)$$

where ξ' lies between ξ_i and ξ_{i-1}.

We see that in numerical grid generation choosing the transformations amounts in essence to choosing the partial derivatives (ξ_x and η_y). This can be done in many ways, as long as the integral (as in eq.(4.2.3)) of the partial derivative is constant and the partial derivative itself is nonzero everywhere.

For example, if more grid points are wanted in those regions where the gradient of the solution is large, we take the partial derivatives proportional to the absolute value of the gradient:

$$\xi_x = \alpha\cdot\left|\frac{\partial\,\omega}{\partial\,x}\right| + \beta \tag{4.2.6}$$

α is determined by the integral in eq.(4.2.3), β is added to ascertain that $\xi_x > 0$ everywhere.

I will return to the choice of the transformation in the next section.

In the following, I will assume that in the physical grid for each grid point l the coordinates x_l, y_l, ξ_l and η_l are known, plus the values of the partial derivatives ξ_x, ξ_y, ξ_{xx}, ξ_{yy}, ξ_{xy} and η_x, η_y, η_{xx}, η_{yy}, η_{xy} are known. I defer to the next section the explanation of how the partial derivatives can be found.

Before a differential equation, for instance the vorticity diffusion equation, can be solved in the computational grid it has also to be transformed . With given partial derivatives of the transformation, the partial derivatives of the equation transform as:

$$\frac{\partial}{\partial\,x} = \xi_x\,\frac{\partial}{\partial\,\xi} + \eta_x\,\frac{\partial}{\partial\,\eta} \tag{4.2.7}$$

$$\frac{\partial}{\partial\,y} = \xi_y\,\frac{\partial}{\partial\,\xi} + \eta_y\,\frac{\partial}{\partial\,\eta} \tag{4.2.8}$$

$$\frac{\partial^2}{\partial\,x^2} = \xi_{xx}\,\frac{\partial}{\partial\,\xi} + \xi_x^2\,\frac{\partial^2}{\partial\,\xi^2} + \eta_{xx}\,\frac{\partial}{\partial\,\eta} + \eta_x^2\,\frac{\partial^2}{\partial\,\eta^2} + 2\cdot\xi_x\eta_x\,\frac{\partial^2}{\partial\,\xi\partial\,\eta} \tag{4.2.9}$$

$$\frac{\partial^2}{\partial y^2} = \xi_{yy}\frac{\partial}{\partial \xi} + \xi_y^2\frac{\partial^2}{\partial \xi^2} + \eta_{yy}\frac{\partial}{\partial \eta} + \eta_y^2\frac{\partial^2}{\partial \eta^2} + 2\cdot\xi_y\eta_y\frac{\partial^2}{\partial \xi \partial y} \tag{4.2.10}$$

$$\frac{\partial}{\partial t} = \xi_t\frac{\partial}{\partial \xi} + \eta_t\frac{\partial}{\partial \eta} + \tau_t\frac{\partial}{\partial \tau} \tag{4.2.11}$$

I assume that the velocity of the grid is negligible and $\tau = t$. ξ_t and η_t are therefore zero and the partial derivative with respect to time transforms into itself. This leads to an error in the solution. This error can be minimised by choosing the time steps small enough.

The vorticity diffusion equation (VDE) becomes transformed:

$$\frac{\partial \omega}{\partial t} = Q(x,y) + \left[-u'\xi_x - v'\xi_y + \nu\xi_{xx} + \nu\xi_{yy}\right]\cdot\frac{\partial \omega}{\partial \xi} +$$

$$+ \left[-u'\eta_x - v'\eta_y + \nu\eta_{xx} + \nu\eta_{yy}\right]\cdot\frac{\partial \omega}{\partial \eta} + \left[\nu\xi_x^2 + \nu\xi_y^2\right]\cdot\frac{\partial^2\omega}{\partial \xi^2} +$$

$$+ \left[\nu\eta_x^2 + \nu\eta_y^2\right]\frac{\partial^2\omega}{\partial \eta^2} + 2\cdot\nu\cdot\left[\xi_x\eta_x + \xi_y\eta_y\right]\cdot\frac{\partial^2\omega}{\partial\xi\partial\eta} \tag{4.2.12}$$

where $Q(x,y)$ is the source term; this is a scalar and isn't changed by the transformation. u' and v' are the transformed velocities:

$$u' = \frac{\partial \Psi}{\partial y} = \xi_y\frac{\partial \Psi}{\partial \xi} + \eta_y\frac{\partial \Psi}{\partial \eta} \tag{4.2.13}$$

$$v' = -\frac{\partial \Psi}{\partial x} = -\xi_x\frac{\partial \Psi}{\partial \xi} - \eta_x\frac{\partial \Psi}{\partial \eta} \tag{4.2.14}$$

If Q, u', v' and the partial derivatives of ξ and η are known, the coefficients of the partial derivatives of ω can be calculated. Then the equation can be substituted by a differential scheme and solved in the same way as described in chapter 2, but now in the (ξ,η) grid. Because in the transformed equation (4.2.12) a term with the mixed derivative $\frac{\partial^2}{\partial\xi\partial\eta}$ appears, eq.(2.13) also has to be used in the differential scheme. The solution of the set of equations then needs a nine-point molecule difference scheme, because the values of ω in the grid points with index l-m-1, l-m+1, l+m-1 and l+m+1 are also included. In [ref.5], a fast solution method for a nine-point molecule difference scheme is introduced.

4.3 Application of grid generation to the actuator strip.

In this section I describe the practical realisation of the numerical adaptive grid, used to calculate the flow through the actuator strip.
The first choice was to let ξ only depend on x, this means that lines of constant ξ are allways straight lines and parallel to the y-axis if drawn in the x-y plane. η is a function of both x and y.
Furthermore, the boundaries of the physical and the computational grid have been chosen the same.
The grid is set up in the following way:
first the vorticity diffusion equation is solved in a regular grid, the solution $\omega(x,y)$ serves as a base for the set-up of the new grid.

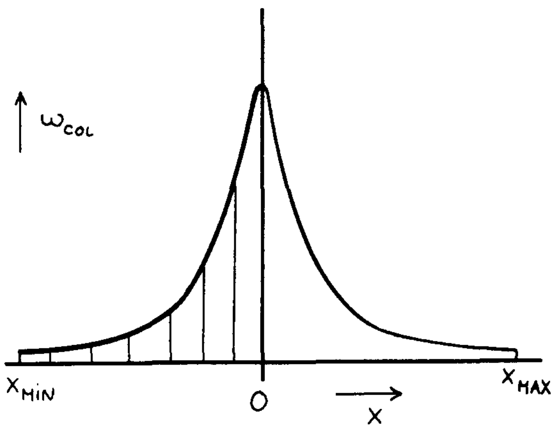


Fig.(4.3.1): The sum of vorticity in a column of grid points as a function of x.

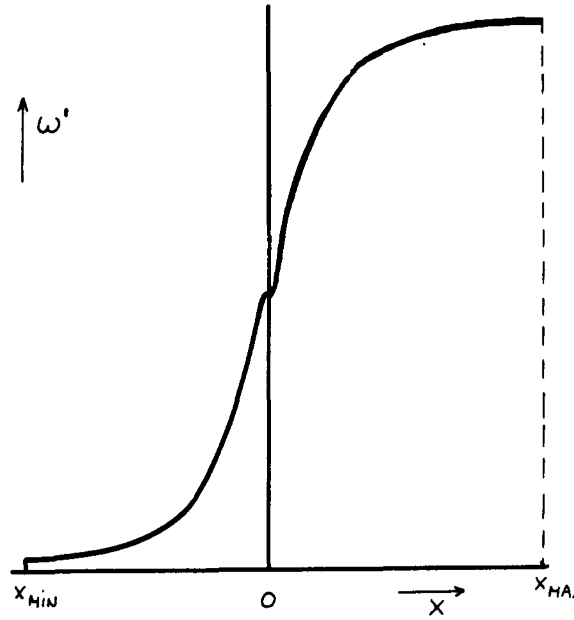


Fig.(4.3.2): Same figure, but now with only positive gradient.

The vorticities in the grid points in each column i ($x = x_i$ = constant) are made positive and added, giving an $\omega_{col,i}$ for each i. This ω_{col} is not a measure for vorticity, but only a quantity that relates the vorticity with the grid. The values of ω_{col} are now altered in such a way, that the differences are allways positive, see figs.(4.3.1) and (4.3.2). The difference between every two consequent ω_{col} is calculated and negated if negative. In this way, the consequent ω_{col} are allways increasing. Simultaneously the difference is altered if it's too small or too large; in other words: in the whole domain the partial derivative of ξ with respect to x is checked. This ξ_x has to lie between a KSIXMIN and a KSIXMAX, the last two are given by the user, but both have to be positive.The altered values of ω_{col} are called ω'.

After this, the coupling between ξ and the ω' is made:

$$\xi_i = A\cdot\omega_i' + B \qquad (4.3.1)$$

A and B are determined in such a way that ξ ranges from XMIN to XMAX.

If we take the derivative of equation (4.3.1) with respect to x we get:

$$\frac{\partial\,\xi_i}{\partial\,x} = A\cdot\frac{\partial\,\omega_i'}{\partial\,x} = A\cdot\,\frac{\partial|\omega_{col\,i}|}{\partial\,x} \qquad (4.3.2)$$

stated in words: the density of grid points in x-direction is proportional to the absolute value of the gradient of the vorticity summed over a column. This means, that the grid points are placed where they are needed the most: in those regions where the gradient in the solution is large.

η and the vorticity are coupled in a similar manner. There are, however,a few differences, which will be discussed in the following.

The expected solution of the problem is a vortex sheet that travels in positive x-direction.(see section 4.1). At the vortex sheet, the vorticity has a local maximum. Because we want to have more grid points in the neighbourhood of these maxima a function g(x) is constructed that connects the local maxima, so g(x) follows the shape of the vortex sheet. g(x) is constructed as follows: for every x_i, the y-coordinate of the maximum of ω is calculated. g(x) is fitted to the y-coordinate of these maxima by a least-squares fit procedure. g(x) is written as the truncated series of D·(1-tanh(x)); if the truncation begins at the NPAR +1rst term, g(x) reads:

$$g(x) = D\cdot\left[1 - \sum_{i=1}^{NPAR} \frac{2\cdot PAR(i)\cdot x}{\pi^2(2\cdot i-1)^2/4 + x^2}\right] \qquad (4.3.3)$$

NPAR is the number of terms in the series, PAR(i) is an array containing the values of the parameters for each term. The value of

PAR(i) is set by the least-squares fit procedure. If NPAR is infinite and PAR(i) = 1 for every i, $g(x) = D\cdot(1 - \tanh(x))$.

Figure (4.3.3) gives a possible shape for g(x). Note that g(x) is point-symmetrical in x = 0, y = D.

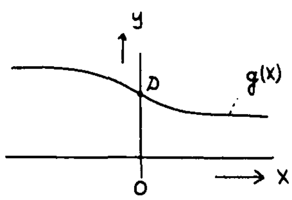


Fig.(4.3.3): Shape of g(x), the line connecting local maxima in the vorticity distribution.

η is coupled to the vorticity in such a way that the number of gridpoints below the line y = g(x) is equal to the number of grid points above the line y = g(x). So y= g(x) is transformed to the line η = YMAX/2.

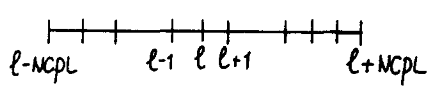


Fig.(4.3.4): The vorticity in NCPL neighbouring grid points is added to the vorticity in each grid point.

In order to give the lines η = constant in the physical grid a smooth course as a function of x, the vorticity in one grid point is increased with the vorticity in neighbouring points (same row, other columns), according to:

$$\omega_l = \left[\sum_{i=1}^{NCPL} \frac{\omega_{l+i}}{(i+1)^2} + \sum_{i=-1}^{-NCPL} \frac{\omega_{l+i}}{(-i+1)^2} \right]. \qquad (4.3.4)$$

The larger NCPL (N couple), the smoother η = constant will be.

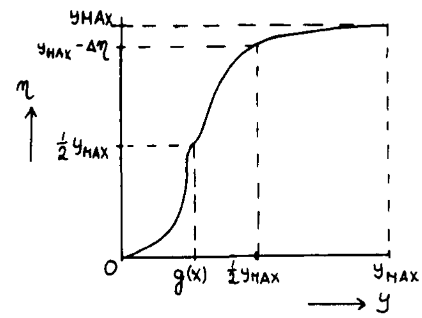


Fig.(4.3.5): Relation between η and y. The values in four points are fixed.

For the coupling the domain is split into two parts, above and below y= g(x). In each column, four transformation values are fixed:

$$\left.\begin{aligned} &\eta(x,0) = 0 \\ &\eta(x,g(x)) = YMAX/2 \\ &\eta(x,YMAX/2) = YMAX - \Delta\eta \\ &\eta(x,YMAX) = YMAX \end{aligned}\right| \qquad (4.3.5)$$

These values are used to determine the coefficients $\alpha_{1,i}$, $\alpha_{2,i}$, $\beta_{1,i}$ and $\beta_{2,i}$ in the same way as A and B were determined in eq.(4.3.1):

Per column $\omega_{i,j}$ is turned into $\omega'_{i,j}$, with the latter only increasing for increasing y and $\frac{\partial \omega'}{\partial y}$ lying between a minimum and a maximum value (ETYMIN and ETYMAX). For $y \leq g(x)$:

$$\eta(x_i,y_j) = \alpha_{1,i} \cdot \omega'_{i,j} + \beta_{1,i} \qquad (4.3.6)$$

and $y \geq g(x)$:

$$\eta(x_i,y_j) = \alpha_{2,i} \cdot \omega'_{i,j} + \beta_{2,i} \qquad (4.3.7)$$

Again α and β are chosen in such a way that η ranges between 0 and YMAX/2 for α_1, β_1 and between YMAX/2 and YMAX for α_2 and β_2.

We see that the columns with i = 1, 2, (m+1)/2, m-1 and m have always x-values XMIN,XMIN/2,0,XMAX/2 and XMAX respectively and that the rows with j = 1, (n+1)/2, n-1 and n have y-coordinates 0, g(x), YMAX/2 and YMAX. Figure (4.3.6) illustrates this.

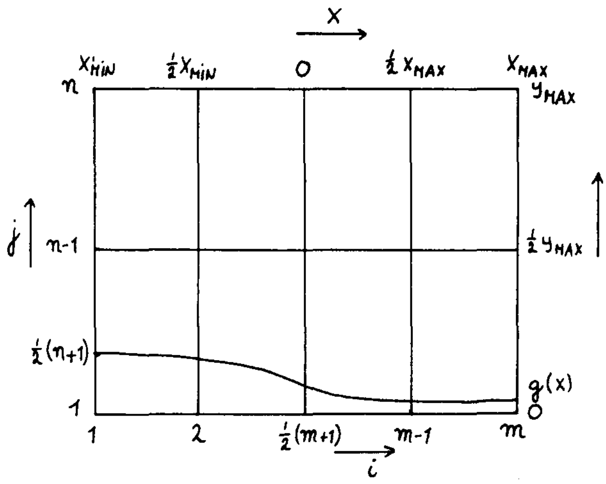
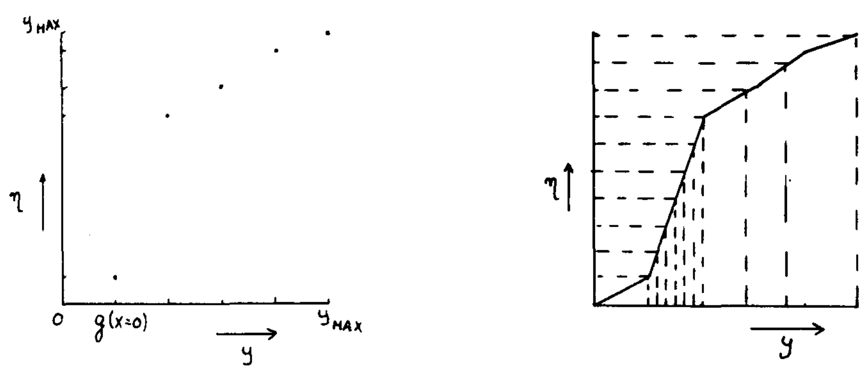


Fig.(4.3.6): ξ = const. and η = constant lines in the physical grid that are held fixed.

If the transformations of the individual grid points are known, the x- and y-coordinates of the new physical grid can be calculated. Figure (4.3.7) gives the relation between old y-coordinates (e.g. for x = 0) and the new, computed η-coordinates. In figure (4.3.8) these points are connected by straight line pieces. Here we see that the y-values of the new physical grid belonging to a regular computational grid are interpolated from the η-values of the old grid.



Figs.(4.3.7) and (4.3.8): Method for calculation of new gridpoints. The solution in a regular physical grid are connected by straight line pieces. The new y-values of a regular computational grid are found by interpolation.

This interpolation is done m+1 times: once for the ξ-x coordinates and m times for the η-y coordinates.

If the new grid is set up, some things have to be done before the calculation of the Poisson-PSI equation in the new grid can begin.
First, the vorticity in the new grid points has to be interpolated from the vorticity in the old points.
Second, the partial derivatives of ξ and η with respect to x and y plus higher order derivatives have to be determined for each grid point.
If 0 is the index for the grid points whose derivatives have to be calculated and 1 the index for a neighbouring point, Taylor development in 0 gives:

$$\eta_1 = \eta_0 + (x_1-x_0)\left.\frac{\partial \eta}{\partial x}\right|_0 + (y_1-y_0)\cdot\left.\frac{\partial \eta}{\partial y}\right|_0 + 1/2\cdot(x_1-x_0)^2\left.\frac{\partial^2\eta}{\partial x^2}\right|_0 +$$

$$+ 1/2\cdot(x_1-x_0)^2\left.\frac{\partial^2\eta}{\partial y^2}\right|_0 + (x_1-x_0)\cdot(y_1-y_0)\cdot\left.\frac{\partial^2\eta}{\partial x\partial y}\right|_0 +\ldots \qquad (4.3.8)$$

If we write down the Taylor series for four other points (index 2-5) the η_x, η_{xx}, η_y, η_{yy} and η_{xy} can be solved from these five equations because η_1-η_5, x_1-x_5 and y_1-y_5 are known. The same is done for the partial derivatives of ξ. This means that for each grid point two sets of five equations, each set with five unknowns, have to be solved. However, ξ is a function of x only, so ξ_y, ξ_{yy} and ξ_{xy} are zero everywhere. Furthermore ξ_x and ξ_{xx} are the same for every grid point in one column; this reduces the number of equations to be solved considerably.

5. The (finite) actuator strip; Calculations and results.

5.1 Equations, boundary- and starting conditions.

The transformed differential equations (eqs. 4.2.12-4.2.14) become more simplified because ξ is a function of x only. The transformed vorticity diffusion equation reads:

$$\frac{\partial \omega}{\partial t} = Q(x,y) + \left[-u'\xi_x + \nu\xi_{xx}\right]\cdot\frac{\partial \omega}{\partial \xi} + \left[-u'\eta_x - v'\eta_y + \nu\eta_{xx} + \nu\eta_{yy}\right] +$$

$$+ \nu\xi_x^2 \frac{\partial^2\omega}{\partial \xi^2} + \left[\nu\eta_x^2 + \nu\eta_y^2\right]\frac{\partial^2\omega}{\partial \eta^2} + 2\cdot\nu\cdot\xi_x\eta_x\frac{\partial^2\omega}{\partial\xi\partial\eta} \qquad (5.1.1)$$

The transformed Poisson-PSI equation is:

$$-\omega = \xi_{xx}\frac{\partial \psi}{\partial \xi} + \xi_x^2 \frac{\partial^2\psi}{\partial \xi^2} + \left[\eta_{xx} + \eta_{yy}\right]\frac{\partial \psi}{\partial \eta} + \left[\eta_x^2 + \eta_y^2\right]\frac{\partial^2\psi}{\partial \eta^2} +$$

$$+ 2\cdot\xi_x\eta_x\frac{\partial^2\psi}{\partial\xi\partial\eta} \qquad (5.1.2)$$

and the transformed velocities:

$$u' = \eta_y\cdot\frac{\partial \psi}{\partial \eta} \qquad (5.1.3)$$

$$v' = -\xi_x\cdot\frac{\partial \psi}{\partial \xi} - \eta_x\cdot\frac{\partial \psi}{\partial \eta} \qquad (5.1.4)$$

The vorticity diffusion equation is solved on the old transformed grid, then a new grid transformation is set up on the basis of the found solution. On the new transformed grid the Poisson-PSI equation is solved and from the streamfunction the velocities are computed.

If the end time isn't reached and again a time step has to be taken, the loop of solving equations starts from the beginning.

The boundary conditions also have to be transformed. These conditions are straightforward for the boundaries x = XMIN, x = XMAX and y = YMAX; here $\omega = 0$ on the boundary. Because u' and v' are taken zero on the boundary, the streamfunction ψ on, as well as next to the boundary is zero.

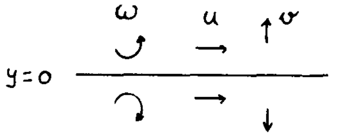


Fig.(5.1.1): ω , u and υ reflected in the boundary y = 0.

The conditions on y = 0 are different. This boundary acts as a mirror, in order to simulate the source at y = -D. See fig.(5.1.1). As can be seen in this figure, the vorticity below is minus the vorticity above the line y = 0. The same holds for v, the velocity in y-direction. The velocity in x-direction, u, doesn't change when crossing y = 0.

The foregoing leads to $\omega = 0$, v = 0 and $\frac{\partial u}{\partial y} = 0$ on the boundary.

$\frac{\partial u}{\partial y} = 0$ leads to:

$$\eta_{yy}\frac{\partial \psi}{\partial \eta} + \eta_y^2 \cdot \frac{\partial^2 \psi}{\partial \eta^2} = 0, \qquad \eta = 0 \qquad (5.1.5)$$

As a consequence of this mirror boundary condition, a whole class of solutions to the equations are excluded. This is for example a solution with asymmetrical vortex shedding from the edges of the strip.

The starting conditions on t = 0 are $\omega = 0$, u = 0, v = 0 and $\psi = 0$ on the whole domain.

The force density on the actuator strip is taken 1 (N/m^3). The time step Δt is used as parameter to guide the tempo of vorticity production at the edge of the strip. The gradient of the force density at the edge of the strip can be altered by giving m in eq.(1.3) another value. This has the effect of broadening or tightening the peak in the vorticity production at the edge of the actuator strip.

5.1 Flow chart, starting grid.

Figure (5.2.1) gives the flow chart of the numerical calculation of the flow through the finite actuator strip. Each square contains a separate action done by a subroutine. The name of the subroutine is placed between brackets. More information about these subroutines can be found in Appendix A2. The whole calculation can be directed by a set of calculation parameters, which are stored in a separate file. The use of these parameters makes it possible for instance first to calculate until t = 5 seconds, then to view the results.

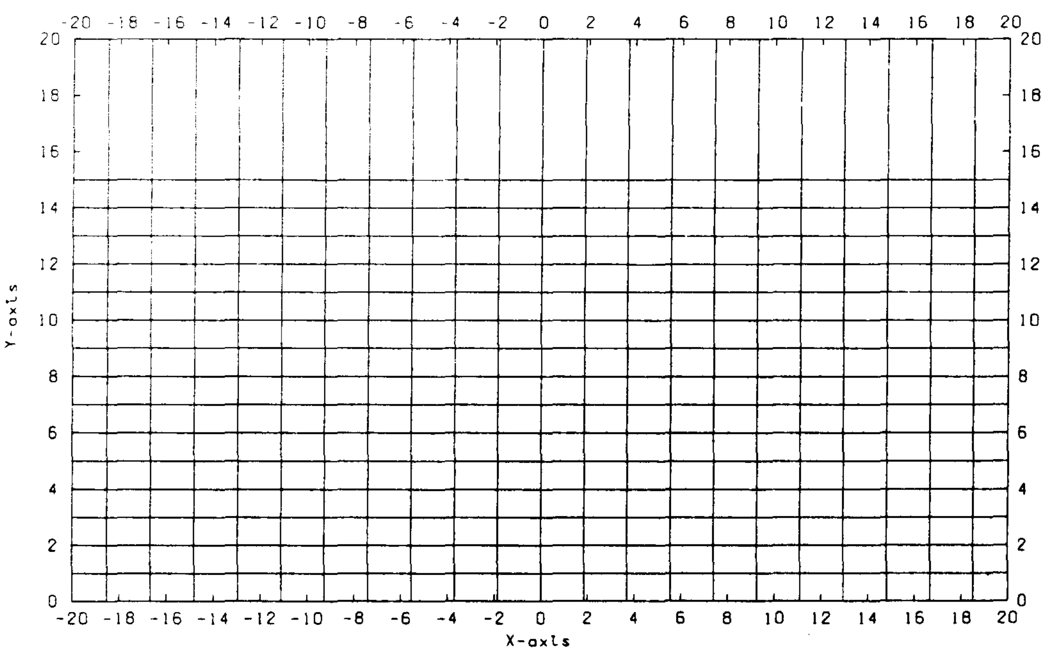


Fig.(5.2.2): Part of the starting grid with 41 by 21 grid points.

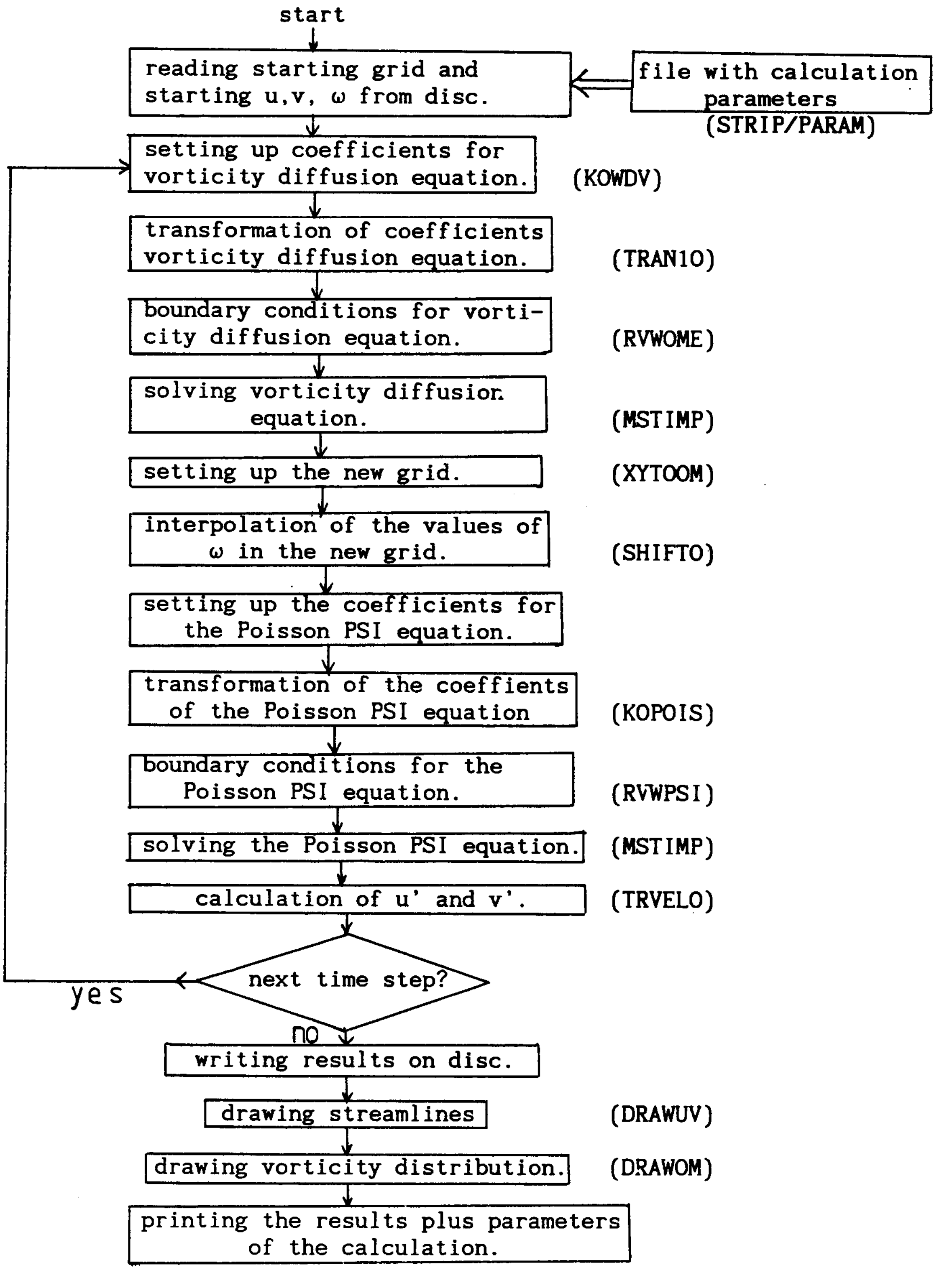


Fig.(5.2.1): Flow chart of the calculation of the flow through the finite actuator strip in the programme STRIP/DEVELOP.

If the results are satisfactory, a second calculation can be performed with the results of the first calculation as starting values.
Figure (5.2.2) gives the starting grid of 21 by 21 points. This grid is set up by the subroutine STAROP. Figure (5.2.3) gives a perspective view of the vorticity distribution at t = 0.1 seconds, calculated in the starting grid.

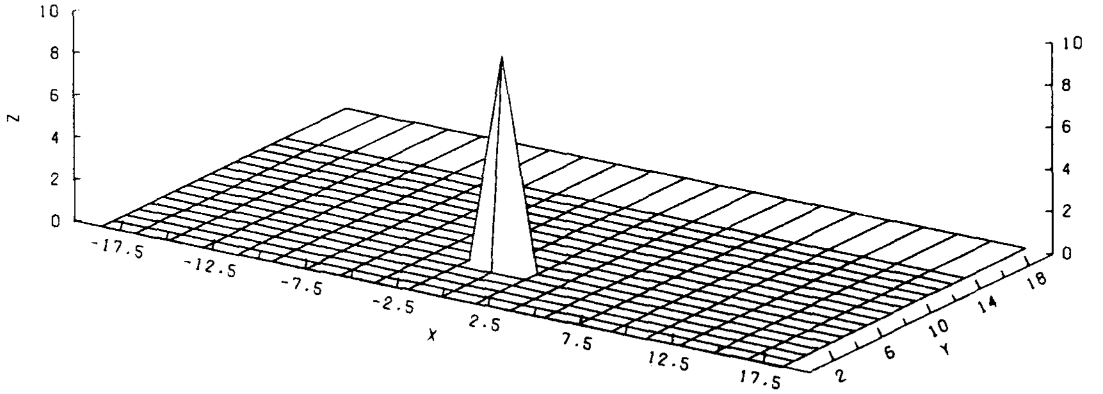


Fig.(5.2.3): Vorticity distribution in a 41 by 21 point regular grid at t=0.1 seconds

In figure (5.2.4), the physical grid is drawn, that results from the grid generation subroutine XYTOOM. This 41 by 21 point grid is adapted to the vorticity distribution, shown in fig.(5.2.3).

The last figure, (5.2.5) gives the vorticity distribution in the adapted grid. This distribution results from linear interpolation of the vorticity distribution in fig.(5.2.3).

The results of calculations with the complete programme will be written in an addendum.

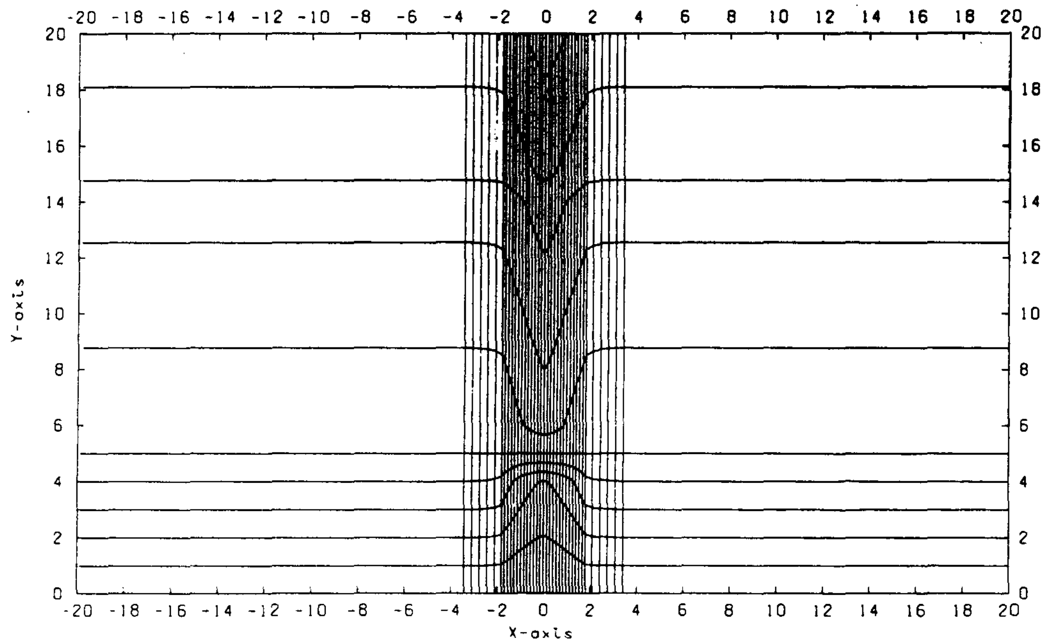


Fig.(5.2.4): Physical grid adapted to the vorticity distribution in the previous figure

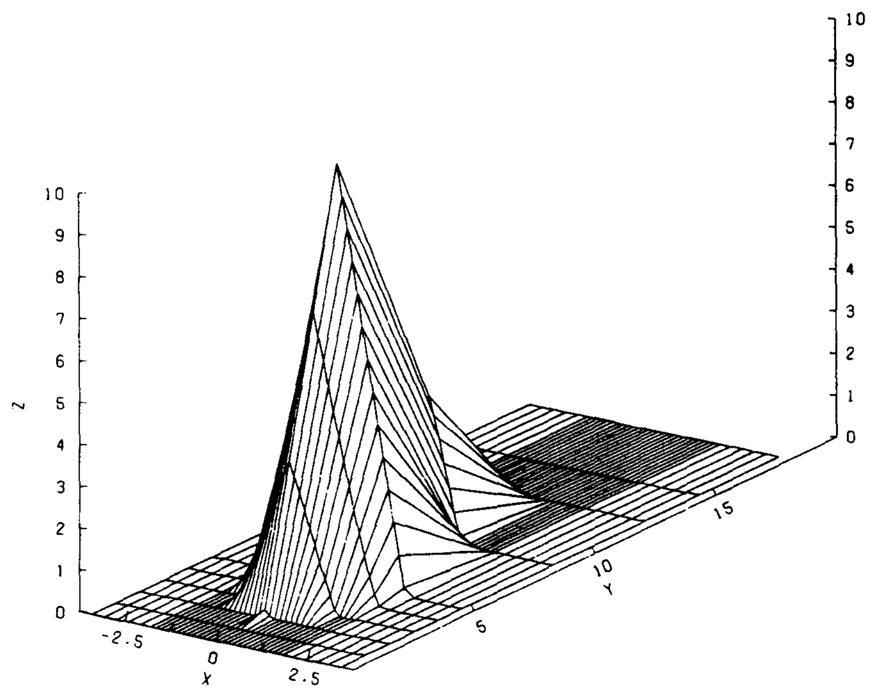


Figure (5.2.5): Vorticity distribution in the adapted grid.

6. Conclusions and Discussion.

The following conclusions can be drawn with respect to the calculation of the flow through the half infinite actuator strip.

- The calculated results conform to the theoretical expected situation; the vorticity, produced at the edge of the strip, is only transported as a result of diffusion and increases as a function of time. The streamlines are concentric circles around the edge of the strip.
- Only in regions near the edge of the strip the value of the vorticity is larger than the machine accuracy ($\approx 1.2 \cdot 10^{-12}$). In spite of the fact that in the rest of the domain the value of the vorticity is smaller than this accuracy, the streamlines found here are also concentric circles.
- The convergence rates found for the Poisson-PSI equation (velocities) are good. However, the convergence rates found for the vorticity diffusion equation, are not as good as could be expected from the order of the method, possibly because too few gridpoints are involved in the calculation of the convergence rates.
- Both solid core source and $\cosh^2$ source cause a velocity field that resembles the velocity field of a point vortex, in regions not close to the source.

When writing this report, the programme used to determine the flow through the finite actuator strip was still in the phase of debugging. The following conclusions are drawn from the results of several testing programmes.

- The adaptation of the grid to the vorticity distribution is satisfactory, but the computing time needed for the grid generation is relatively long; for a 41x21 point grid the computation takes $\approx$ 4s, the grid generation $\approx$13s.
- The block structure of the programme facilitates changes in parts of the programme. For instance, it's relatively easy to build in an undisturbed velocity Uo.

LIST OF REFERENCES.

1. Patrick J. Roache, "Computational Fluid Dynamics"
Hermosa Publishers, Albuquerque. 1976

2. J.A. Sparenberg, "Elements of Hydrodynamic Propulsion."
Martinus Nijhoff Publishers. 1984

3. H.L. Stone, "Iterative Solutions of Implicit Approximations of Multidimensional Partial Differential Equations".
SIAM J.Numer.Anal. 5 nr 3 sept'68 p530-

4. D.A.H. Jacobs, "The Strongly Implicit Procedure for Biharmonic problems."
J.of Comp.Phys. 13, 303-315 (1973).

5. G.E. Schneider & M. Zedan,"A Modified Strongly Implicit Procedure for the Numerical Solution of Field Problems."
Numerical Heat Transfer, vol.4, pp 1-19, 1981

6. G.A.M. van Kuik,
PhD. Thesis, to be published in the course of 1988,
Eindhoven University of Technology, Dept. of Physics.

7. Z.U.A. Warsi & J.F. Thompson,"The Importance of Dynamically Adaptive Grids in the Numerical Solution of Partial Differential Equations."
von Karman Institute for Fluid Dynamics. Lecture series 1984-04. Computational Fluid Dynamics.
March 12-16, 1984.

8.J.F. Thompson, "General Curvilinear Coordinate Systems."
in:"Numerical Grid Generation.",
editor Joe F. Thompson,
Elsevier Science Publishing Company Inc.1982.

9. Anderson, Tannehill & Pletcher,"Computational Fluid Dynamics and Heat Transfer."
Hemisphere,1984-XII

10. G.H. Schmidt & J.A. Sparenberg,"On the Edge Singularity of an Actuator Disk with Large Constant Normal Load."
J.of Ship Research, Vol.21 nr.2 june 1977 pp125-131

11. M.D. Greenberg & J.H.W. Lee,"Line Momentum Source in Shallow, Inviscid Fluid."
J.of Fluid Mechanics, Vol.145, 1984

12. G.A.M. van Kuik, "The Actuator Disc Edge Singularity: the Key to a Revised Actuator Disc Concept and Momentum Theory."
paper no.2-16, 13th European Rotorcraft Forum, sept.8-11, 1987, Arles, France.

13. A.M. van der Spek, "Linear and Non-linear Theories of Flow Through an Actuator Disc.
Master's Thesis, Eindhoven Technical University, Dept. of Physics, report no. R-803-A.

14. T.Y. Wu, "Flow Through a Heavily Loaded Actuator Disc."
Schiffstechnik, Bd.9-1962, Heft 47.

15. H.S. Carslaw and J.C. Jaeger,"Conduction of Heats in Solids."
Clarendon Press, Oxford, 1986.

16. M. Abramowitz and I.A. Stegun,"Handbook of Mathematical Functions."
Dover Publications,I Inc., New York 1965.

Appendix A1

De sterk impliciete methode van Stone.[lit.3]

Dit is een methode voor het oplossen van grote stelsels vergelijkingen, die geschreven kunnen worden als:

$$S\ T = q \qquad (1)$$

dit is een stelsel vergelijkingen, dat verkregen wordt door bijvoorbeeld een tweede orde partiele differentiaalvergelijking te diskretiseren in een rooster van twee dimensies.
Voor een rooster met MxN punten is S een vierkante matrix met (MxN)^2 elementen.
S is pentadiagonaal in het geval van een 5-punts diskretisatie, hier verder het geval.
S ziet er als volgt uit:

Uitgeschreven voor een roosterpunt ziet de vergelijking er als volgt uit:

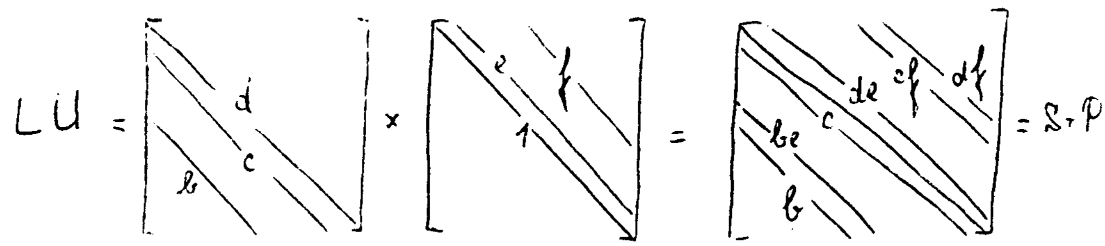
$$q_{j,k} = B_{j,k}T_{j,k-1} + D_{j,k}T_{j-1,k} + D_{j,k}T_{j,k} + F_{j,k}T_{j+1,k} + H_{j,k}T_{j,k+1}$$

j is de index voor de x-koordinaat van een roosterpunt, k de index voor de y-koordinaat.
Een manier om het stelsel (1) op te lossen is om de matrix S te schrijven als het produkt van twee matrices L en U, met L een linksonderdriehoeksmatrix en U een rechtsbovendriehoeksmatrix.
In het algemeen kost deze zogenaamde LU-dekompositie veel rekenwerk, omdat de matrix S meestal geen ijle matrices L en U oplevert. Onder een ijle matrix wordt verstaan een matrix, waarvan alle rijen op enkele na met nullen gevuld zijn.

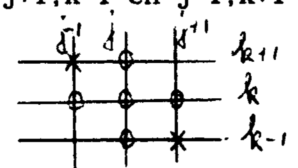
Willen we een ijle L en U vinden, bijvoorbeeld L en U beiden tridiagonaal, dan moet bij S een matrix P opgeteld worden met P zodanig dat:

$$S + P = LU \qquad (2)$$

Om te zien aan welke voorwaarden P moet voldoen, schrijven we LU uit:



S + P is een heptadiagonale matrix, de twee ten opzichte van S toegevoegde diagonalen kunnen gezien worden als koefficienten, horend bij de roosterpunten j+1,k-1 en j-1,k+1 (kruisjes in figuur a1).



Deze koefficienten noemen we $C_{j,k}$ en $G_{j,k}$ respektievelijk.
De funktiewaarden in de punten j+1,k-1 en j-1,k+1 kunnen geschreven worden als een lineaire kombinatie van de funktiewaarden in de drie naastliggende punten. Dit kan door Taylorreeksontwikkeling met weglating van termen van de ordegrootte x^3, y^3, en x y. Dan geldt:

$$T_{j-1,k+1} = -T_{j,k} + T_{j,k+1} + T_{j-1,k} \qquad (3)$$

en:

$$T_{j+1,k-1} = -T_{j,k} + T_{j+1,k} + T_{j,k-1} \qquad (4)$$

Om de koefficienten $C_{j,k}$ en $G_{j,k}$ zo klein mogelijk te houden (S+P lijkt dan veel op S) kan het beste een ongeveer even grote waarde van $T_{j-1,k+1}$ en $T_{j+1,k-1}$ worden afgetrokken.

Het differentieschema voor S + P wordt dan:

$$B_{j,k}T_{j,k-1} + D_{j,k}T_{j-1,k} + E_{j,k}T_{j,k} + F_{j,k}T_{j+1,k} + H_{j,k}T_{j,k+1} +$$

$$+ C_{j,k}[\, T_{j+1,k-1} - \alpha(-T_{j,k} + T_{j+1,k} + T_{j,k-1})] +$$

$$+ G_{j,k}[\, T_{j-1,k+1} - \alpha(-T_{j,k} + T_{j-1,k} + T_{j,k+1})] = q_{j,k} \qquad (5)$$

Gelijkstelling van de koefficienten van S+P en LU levert:

$$b_{j,k} = B_{j,k} - \alpha C_{j,k} \qquad (6)$$

$$c_{j,k} = D_{j,k} - \alpha G_{j,k} \qquad (7)$$

$$d_{j,k} + b_{j,k}f_{j,k-1} + c_{j,k}e_{j-1,k} = E_{j,k} + \alpha C_{j,k} + \alpha G_{j,k} \qquad (8)$$

$$d_{j,k}e_{j,k} = F_{j,k} - \alpha C_{j,k} \qquad (9)$$

$$d_{j,k}f_{j,k} = H_{j,k} - \alpha G_{j,k} \qquad (10)$$

Uit deze vergelijkingen zijn iteratief de onbekenden $b_{j,k}$, $c_{j,k}$, $d_{j,k}$, $e_{j,k}$ en $f_{j,k}$ op te lossen. Op de koefficient α komen we later terug.
Het bepalen van de koefficienten kan op twee manieren:

1) j oplopende van 1 tot M, k oplopend van 1 tot N
2) j weer oplopend, k afnemend van N tot 1, waarbij in de vergelijkingen (6) t/m (10) de j+1 wordt vervangen door j-1 en andersom.

De tweede manier komt neer op het doorlopen van het rooster van linksboven naar rechtsbeneden.
Er wordt steeds gebruik gemaakt van het feit dat de funktie buiten het rooster de waarde nul heeft:

$$c_{1,1} = 0 \quad \text{en} \quad C_{1,1} = 0 \qquad (11)$$

$$b_{j,1} = 0 \quad \text{en} \quad B_{j,1} = 0 \qquad (12)$$

$$f_{j,N} = o \quad \text{en} \quad H_{j,N} = 0 \qquad (13)$$

$$F_{M,k} = 0 \qquad (14)$$

Als de elementen van L en U bepaald zijn, kan het stelsel vergelijkingen (1) worden opgelost. In vergelijking (1) wordt aan beide zijden PT opgeteld:

$$(S + P)T = NT + q \tag{15}$$

Dit is ook gelijk aan:

$$(S + P)T = (S + P)T - (ST - q) \tag{16}$$

Deze vergelijkingen worden gebruikt voor het oplossen van het stelsel (1); stel $T^{(n-1)}$ is de gevonden oplossing na de n-de iteratie; de nieuwe oplossing is dan gelijk aan:

$$(S + P)T^{(n)} = (S + P)T^{(n-1)} - (ST^{(n-1)} - q) \tag{17}$$

met $\delta^{(n)} = T^{(n)} - T^{(n-1)}$ en $R^{(n-1)} = q - ST^{(n-1)}$ wordt het op te lossen stelsel:

$$(S + P)\delta^{(n)} = R^{(n-1)} \tag{18}$$

of:

$$LU\delta^{(n)} = R^{(n-1)} \tag{19}$$

Om tot een snellere konvergentie van de oplossing te komen kan de koefficient α uit de vergelijkingen (6) t/m (10) het beste worden gevarieerd. De maximale waarde van α volgt volgens Jacobs [lit.4] uit de afmetingen van het rooster:

$$\alpha_{max} = 1- \frac{2}{(M-1)^2+(N-1)^2} \tag{20}$$

Voor elke iteratie wordt een nieuwe α genomen volgens

$$\alpha_i = 1 - (1 - \alpha_{max})^{(i-1)/8} \qquad i=1,9 \tag{21}$$

De volgorde waarin i wordt verwisseld is 9,6,3,8,5,2,7,4,1. Op deze manier omspant α_i een groot bereik tussen 0 en α_{max} binnen een groepje van 3 opeenvolgende i.

WEG/WAM/STRIP/HO
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 12:45:15.

```
      FILE  6(KIND='REMOTE')
      FILE  8(KIND='PRINTER')
          BLOCK GLOBALS
C
C  This program calculates the vorticity distribution, the streamfunc-
C  tion and the velocity field of a half-infinite actuator strip with
C  its edge on the origin.
C  It uses the subroutines KOEFAF, KOEFRP and BEGWRD in the library LIB
C  and GRIDOP, VELOD2 and STIMSY in the library MAINLI.
C  The maximum grid size is 20000 gridpoints.
C
          LIBRARY LIB(TITLE='OBJECT/WEG/WAM/STRIP/HO/STRIP/LIBRARY.')
          LIBRARY MAINLI(TITLE='OBJECT/WEG/WAM/STRIP/HO/MAIN/LIBRARY/
     *    SOURCE.')
       END
       INTEGER I,J,K,L,M,N,NT,NTIT,NX,NY,NVF,TELLER,TELOME,TELPSI,LOOP,
     *          IFALSE,XDELER,YDELER,TSOURC
       REAL A(20000),B(20000),C(20000),D(20000),
     *      E(20000),F(20000),G(20000),
     *      H(20000),KA(20000),P(20000),R(20000),
     *      X(20000),Y(20000),U(20000),
     *      V(20000),PSI(20000),OMO(20000),OMN(20000),
     *      VHULP(20000),OMAKS(20000),
     *      OM1(20000),Q(20000),FAC1,FAC2,CONV,RELRES,
     *      H1(20000),H2(20000),H3(20000),H4(20000),
     *      H5(20000),H6(20000),H7(20000),
     *      H8(20000),FSTR,RHOO,NU,DT,EPS1,EPS2,OMEGA1,
     *      OMEGA2,DEELT,NTQMAX,
     *      NORM,NORMAK,ALFA,AKERN
C
C  The computation is directed by parameters, that are stored in the
C  file 'STRIP/HO/PARAMETERS'.
C
       OPEN(10,FILE='STRIP/HO/PARAMETERS.',STATUS='OLD')
       OPEN(20,FILE='ROTATIEVELD.',STATUS='NEW')
       OPEN(30,FILE='SNELHEIDSVELD.',STATUS='NEW')
       OPEN(40,FILE='OMPSIUV.',STATUS='NEW')
C
C  NT: number of time steps
C  TBEGIN: beginning time of the calculation
C  TEIND: ending time of calculation
C  M: number of grid points in X-direction
C  N: number of grid points in Y-direction
C  FSTR: value of the force on the actuator strip
C  NX: outer boundary of the grid in X-direction
C  NY:   "      "     "  "   "   "  Y-    "
C  NVF: parameter to determine the strength of the non-regular
C       distribution of grid points, used in the subroutine GRIDOP
C  EPS1: accuracy for solving the vorticity diffusion equation
C  OMEGA1: iteration parameter
C  EPS2: same as EPS1, for solving the Poisson-PSI equation.
C  OMEGA2: iteration parameter
C  NTQMAX: time at which the force on the actuator strip is at
C          maximum strength. The strip is turned on at t=0.
C  TSOURC: type of source used to describe the behaviour of the
C          force density at the edge of the strip.
C  AKERN: distance to the core of the source where the force has
C         decreased to about 1/1000 th of its maximum.
C
       RHOO=1.2
       NU=1.45E-5
       READ(10,*)NT,TBEGIN,TEIND,M,N,FSTR,NX,NY,NVF,EPS1,OMEGA1,EPS2,
     * OMEGA2,NTQMAX,TSOURC,AKERN
       WRITE(20,*)NT,TBEGIN,TEIND,M,N,FSTR,NX,NY,NVF,EPS1,OMEGA1,EPS2,
     * OMEGA2,NTQMAX,TSOURC,AKERN
       WRITE(30,*)NT,TBEGIN,TEIND,M,N,FSTR,NX,NY,NVF,EPS1,OMEGA1,EPS2,
     * OMEGA2,NTQMAX,TSOURC,AKERN
       WRITE(40,*)NT,TBEGIN,TEIND,M,N,FSTR,NX,NY,NVF,EPS1,OMEGA1,EPS2,
     * OMEGA2,NTQMAX,TSOURC,AKERN
       L=M*N
C
C  The starting values for the various variables are determined in
C  BEGWRD.
C
       DT=TEIND/NT
       CALL BEGWRD(L,M,N,U,V,OMO,OMN,PSI,DT,0.)
C
C  The coordinates of the grid points are calculated in GRIDOP.
C
       CALL GRIDOP(X,Y,M,N,NVF,NX,NY)
C
C  beginning of the calculation; the loop is repeated for every time
C  step.
C
       DO 200 NTIT=1,NT
C
C  DEELT is smaller than 1 if the force hasn't reached its maximum
C  strenght.
C
       IF(NTIT*DT.GT.NTQMAX)DEELT=1.
       IF(NTIT*DT.LE.NTQMAX)DEELT=NTIT*DT/NTQMAX
C
C  In KOEFAF the coefficients of the discretised vorticity diffusion
C  equation are determined.
C
       CALL KOEFAF(A,B,C,D,E,F,OMO,L,M,N,DT,U,V,X,Y,RHOO,NU,FSTR,
     *             DEELT,AKERN,TSOURC)
```

```
C
C  The vorticity diffusion equation is solved by the strongly implicit,
C  symmetric procedure.
C
      IFALSE=0
      CALL STIMSY(A,B,C,D,E,OMN,F,L,M,N,EPS1,TELOME,H1,H2,H3,H4,H5,H6,
     *              H7,H8,VHULP)
C
C  In KOEFRP the coefficients of the discretised Poisson-PSI equation
C  are determined.
C
      CALL KOEFRP(X,Y,L,M,N,R,G,H,KA,P,Q,OMN)
C
C  Again STIMSY is used, now to solve the Poisson-PSI equation.
C
      CALL STIMSY(R,G,H,KA,P,PSI,Q,L,M,N,EPS2,TELPSI,H1,H2,H3,
     *              H4,H5,H6,H7,H8,VHULP)
C
C  In VELOD2 the velocity components are calculated from the stream-
C  function.
C
      CALL VELOD2(U,V,X,Y,PSI,L,M,N)
C
C  For the calculatiion in the next time step, the values of OMN (omega
C  new) are copied to OMO (omega old).
C
      DO 160 I=1,L
        OMO(I)=OMN(I)
  160 CONTINUE
C
C  If TEIND isn't reached yet, the whole calculation is repeated for
C  the next time step.
C
  200 CONTINUE
      WRITE(6,115)TELOME
  115 FORMAT(1X,'Werveldiffusievergelijking opgelost.',/,1X,'Aantal',
     *          ' verrichte iteraties: ',I4,/)
C
C  The results of the calculation are stored in FILE40 (OMPSIUV.).
C  By using other programmes it's possible to make a plot of the
C  streamlines, of the grid or of the vorticity distribution, or to
C  get a hard copy of the results.
C
      DO 130 I=1,L
 130  WRITE(40,*)X(I),Y(I),OMN(I),PSI(I),U(I),V(I)
C
C  For use in e.g. convergence calculations, the number of grid points
C  is reduced to 9x9 and the values of the vorticity and the velocities
C  are stored in FILE20 and FILE30, respectively.
C
      XDELER=(M-1)/8
      YDELER=(N-1)/8
      DO 120 K=1,9
        DO 120 I=1,9
          OMN(I+(K-1)*9)=OMN((I-1)*XDELER+1+((K-1)*YDELER)*M)
          U(I+(K-1)*9)=U((I-1)*XDELER+1+((K-1)*YDELER)*M)
          V(I+(K-1)*9)=V((I-1)*XDELER+1+((K-1)*YDELER)*M)
  120 CONTINUE
C
C  wegschrijven van de snelheidswaarden voor de laatste tijdstap:
C
      DO 150 K=9-1,0,-1
        WRITE(20,*)(OMN(K*9+I),I=1,9)
        WRITE(30,*)(U(K*9+I),I=1,9)
        WRITE(30,*)(V(K*9+I),I=1,9)
  150 CONTINUE
      WRITE(20,*)TELOME
      WRITE(30,*)TELPSI
      WRITE(40,*)TELOME,TELPSI
C
C
C  afsluiten files
C
      CLOSE(20,STATUS='CRUNCH')
      CLOSE(10,STATUS='CRUNCH')
      CLOSE(30,STATUS='CRUNCH')
      CLOSE(40,STATUS='CRUNCH')
      WRITE(6,300)TELPSI
  300 FORMAT(1X,'Aantal verrichte iteraties bij het oplossen van de',
     *          ' Poisson-PSI vgl:'I4,/)
      END
      SUBROUTINE GRIDOP(X,Y,M,N,NVF,NX,NY)
      INTEGER M,N,NVF,NX,NY
      REAL X(M*N),Y(M*N)
      IN LIBRARY MAINLI
      END
      SUBROUTINE SSOR(A,B,C,D,E,F,V,VH,L,M,EPS,OMEGA,TELLER,IFALSE)
      INTEGER L,M,TELLER,IFALSE
      REAL A(L),B(L),C(L),D(L),E(L),F(L),V(L),VH(L),EPS,OMEGA
      IN LIBRARY MAINLI
      END
      SUBROUTINE KOEFAF(A,B,C,D,E,F,OMO,L,M,N,DT,U,V,X,Y,RHOO,NU,FSTR,
     *                  DEELT,AKERN,TSOURC)
      INTEGER L,M,N,TSOURC
      REAL A(L),B(L),C(L),D(L),E(L),F(L),OMO(L),U(L),V(L),X(M*N),
     *     Y(M*N),DT,RHOO,NU,FSTR,DEELT,AKERN
      IN LIBRARY LIB
      END
      SUBROUTINE KOEFRP(X,Y,L,M,N,R,G,H,KA,P,Q,OMO)
      INTEGER L,M,N
```

```
            REAL X(M*N),Y(M*N),R(L),G(L),H(L),KA(L),P(L),Q(L),OMO(L)
            IN LIBRARY LIB
            END
            SUBROUTINE VELOD2(U,V,X,Y,PSI,L,M,N)
            INTEGER L,M,N
            REAL U(L),V(L),X(L),Y(L),PSI(L)
            IN LIBRARY MAINLI
            END
            SUBROUTINE BEGWRD(L,M,N,U,V,OMO,OMN,PSI,DT,FSTR)
            INTEGER L,M,N
            REAL U(M),V(N),OMO(L),OMN(L),PSI(L),DT,FSTR
            IN LIBRARY LIB
            END
            SUBROUTINE STIMP1(R,G,H,KA,P,PSI,OM,L,M,N,ALFA,EPS,TELPSI,
           *                  A,B,C,D,E,F,VH,RH,DE)
            INTEGER L,M,N,TELPSI
            REAL R(L),G(L),H(L),KA(L),P(L),PSI(L),OM(L),A(L),B(L),C(L),
           *     D(L),E(L),F(L),VH(L),RH(L),DE(L),ALFA,EPS
            IN LIBRARY MAINLI
            END
            SUBROUTINE STIMSY(R,G,H,KA,P,PSI,OM,L,M,N,EPS,TELPSI,
           *                  A,B,C,D,E,F,VH,RH,DE)
            INTEGER L,M,N,TELPSI
            REAL R(L),G(L),H(L),KA(L),P(L),PSI(L),OM(L),A(L),B(L),C(L),
           *     D(L),E(L),F(L),VH(L),RH(L),DE(L),EPS
            IN LIBRARY MAINLI
            END
WEG/WAM/STRIP/HO/MAIN/LIBRARY/SOURCE
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 13:13:08.

      FILE  6(KIND=´REMOTE´)
            BLOCK GLOBALS
              EXPORT(GRIDOP,SSOR,VELOD3,VELOD2,STIMP1,STIMSY)
            END
            CALL FREEZE(´TEMPORARY´)
            END
            SUBROUTINE GRIDOP(X,Y,M,N,NVF,NX,NY)
            INTEGER M,N,NVF,NX,NY,I,J
            REAL X(M*N),Y(M*N)
      C
      C  This subroutine generates a non linear distribution of grid points.
      C  The distance between grid points decreases if the grid points lie
      C  nearer to the coordinate axes. The rate of decrease is determined
      C  lie in the centre of the grid.
      C
            DO 10 J=1,M
               DO 10 I=1,M
                  K=(J-1)*M+I
                  X(K)=NX*((2.*(I-1)/(M-1)-1.)**(2.*NVF+1))
         10       Y(K)=NY*((2.*(J-1)/(N-1)-1.)**(2.*NVF+1))
            RETURN
            END
```

```
            SUBROUTINE VELOD2(U,V,X,Y,PSI,L,M,N)
            INTEGER J,I,K,L,M,N
            REAL KI,KJ,S1,S2,U(L),V(L),X(L),Y(L),PSI(L)
      C
      C  Determination of the velocity components in X- and in Y-direction
      C  from the streamfunction.
      C  Discretisation schemes with truncation errors of order 2 are used.
      C
            DO 100 J=2,N-1
              DO 100 I=2,M-1
                K=I+(J-1)*M
                KI=X(K+1)-X(K-1)
                V(K)=(PSI(K-1)-PSI(K+1))/KI
                KJ=Y(K+M)-Y(K-M)
                U(K)=(PSI(K+M)-PSI(K-M))/KJ
        100 CONTINUE
            DO 200 J=1,N
              DO 200 I=1,M
                K=I+(J-1)*M
                IF (I.EQ.1)THEN
                  KI=X(K+1)-X(K)
                  V(K)=(PSI(K)-PSI(K+1))/KI
                END IF
                IF(I.EQ.M)THEN
                  KI=X(K)-X(K-1)
                  V(K)=(PSI(K-1)-PSI(K))/KI
                END IF
                IF(J.EQ.1)THEN
                  KJ=Y(K+M)-Y(K)
                  U(K)=(PSI(K+M)-PSI(K))/KJ
                END IF
                IF(J.EQ.N)THEN
                  KJ=Y(K)-Y(K-M)
                  U(K)=(PSI(K)-PSI(K-M))/KJ
                END IF
        200 CONTINUE
            RETURN
            END
            SUBROUTINE STIMSY(R,G,H,KA,P,PSI,OM,L,M,N,EPS,TELPSI,A,B,C,
           *                  D,E,F,VH,RH,DE)
            INTEGER L,M,N,J,TELPSI,NT,NITV(9),NIT
            REAL R(L),G(L),H(L),KA(L),P(L),PSI(L),OM(L),A(L),B(L),C(L),D(L),
           *     E(L),F(L),VH(L),RH(L),DE(L),ALFA,EPS,REST,Q,S,ITPAR(9),ALMAX
      C
      C  This subroutine applies the strongly implicit, symmetric method of
      C  Stone to solve a discretised differential equation in two
      C  dimensions.
      C  The equations, to be solved, read:
      C
      C     RiPSIi-m + GiPSIi-1 + HiPSIi + KAiPSIi+1 + PiPSIi+m = OMi
```

```
C
C  This is written in marix form as:
C
C    MA PSI = OM
C
C  PSI is the value to be solved.
C  To MA a sparse matrix NA is added, in such a way that:MA + NA = L U
C  L and U both are triangular.
C
C  The equation then can be written as:
C
C    (MA + NA)PSI' = (MA + NA)PSI + OM - NA PSI
C
C  or:
C
C    L U(PSI'-PSI)= RH
C    L U DE = RH
C    L VH = RH
C    L VH = RH
C  and:
C    U DE = VH
C
C  RH is a  residual vector
C  VH is a help vector
C
C  First a set of iteration parameters ALFA is made, varying between
C  zero and ALMAX.
C
      DO 10 I=3,1,-1
        DO 10 J=0,2
 10       NITV(10-3*I+J)=6+I-3*J
      ALMAX=1.-2./((M-1)**2+(N-1)**2)
      DO 20 I=1,9
 20     ITPAR(I)=1.-(1.-ALMAX)**((I-1.)/8.)
      NT=0
      NIT=0
 50   NIT=NIT+1
      NT=NT+1
      IF(NIT.EO.10)NIT=1
      ALFA=ITPAR(NITV(NIT))
C
C  calculating the residualvector RH(I):
C
      DO 100 J=1,L
        RH(J)=OM(J)-H(J)*PSI(J)
        IF(J.GE.2)RH(J)=RH(J)-G(J)*PSI(J-1)
        IF(J.GE.M+1)RH(J)=RH(J)-R(J)*PSI(J-M)
        IF(J.LE.L-M)RH(J)=RH(J)-P(J)*PSI(J+M)
        IF(J.LE.L-1)RH(J)=RH(J)-KA(J)*PSI(J+1)
100   CONTINUE
C
C  Determination of the coefficients of the L and U matrices, for both
C  increasing row number and column number.
C
      DO 200 J=1,L
        B(J)=0.
        IF(J.GT.M)B(J)=R(J)/(1.+ALFA*E(J-M))
        Q=0.
        IF(J.GT.M)Q=B(J)*E(J-M)
        C(J)=0.
        IF(J.GT.1.AND.J.LE.L-M+1)C(J)=G(J)/(1.+ALFA*F(J-1))
        IF(J.GT.L-M+1)C(J)=G(J)
        S=0.
        IF(J.GT.1.AND.J.LE.L-M+1)S=C(J)*F(J-1)
        D(J)=H(J)+ALFA*(Q+S)
        IF(J.GE.M+1)D(J)=D(J)-B(J)*F(J-M)
        IF(J.GE.2)D(J)=D(J)-C(J)*E(J-1)
        IF(J.EQ.L)E(J)=0.
        IF(J.LT.L)E(J)=(-ALFA*Q+KA(J))/D(J)
        F(J)=-ALFA*S/D(J)
        IF(J.LE.L-M)F(J)=F(J)+P(J)/D(J)
200   CONTINUE
C
C
C  Solution with lower triangular matrix.
C
      DO 300 J=1,L
        IF(J.GE.2)RH(J)=RH(J)-C(J)*VH(J-1)
        IF(J.GE.M+1)RH(J)=RH(J)-B(J)*VH(J-M)
300     VH(J)=RH(J)/D(J)
C
C  Solution with upper trianular matrix; DE(I) is the difference
C  new and old solution.
C
      DO 400 J=L,1,-1
        IF(J.LT.L)VH(J)=VH(J)-E(J)*DE(J+1)
        IF(J.LE.L-M)VH(J)=VH(J)-F(J)*DE(J+M)
400   DE(J)=VH(J)
C
C  Determination of new solution PSI, plus the maximum difference
C  new and old solution.
C
      REST=0.
      DO 500 J=1,L
        PSI(J)=PSI(J)+DE(J)
        IF(ABS(DE(J)).GT.REST)REST=ABS(DE(J))
500   CONTINUE
      IF(REST.LT.EPS)GO TO 1000
      NT=NT+1
```

```
C
C  calculation of the residual vector RH(I)
C
      DO 450 J=1,L
        RH(J)=OM(J)-H(J)*PSI(J)
        IF(J.GE.2)RH(J)=RH(J)-G(J)*PSI(J-1)
        IF(J.GE.M+1)RH(J)=RH(J)-R(J)*PSI(J-M)
        IF(J.LE.L-M)RH(J)=RH(J)-P(J)*PSI(J+M)
        IF(J.LE.L-1)RH(J)=RH(J)-KA(J)*PSI(J+1)
  450 CONTINUE
C
C  determination of the coefficients of the L and U matrices, for
C  decreasing row number and increasing column number.
C
  600 DO 700 K=N,1,-1
        DO 700 I=1,M
          J=I+(K-1)*M
          IF(K.LT.N.AND.K.GT.1)B(J)=R(J)/(1.+ALFA*E(J+M))
          IF(K.EQ.1)B(J)=0.
          IF(K.EQ.N)B(J)=R(J)
          IF(J.EQ.1)C(J)=0.
          IF(I.EQ.1.AND.J.GT.1)C(J)=G(J)
          IF(I.GT.1)C(J)=G(J)/(1.+ALFA*F(J-1))
          D(J)=H(J)
          IF(J.GT.1)D(J)=D(J)-C(J)+G(J)
          IF(I.GT.1)D(J)=D(J)-C(J)*E(J-1)
          IF(K.GT.1)D(J)=D(J)+R(J)-B(J)
          IF(K.GT.1.AND.K.LT.N)D(J)=D(J)-B(J)*F(J+M)
          E(J)=0.
          IF(J.LT.L)E(J)=KA(J)
          IF(K.GT.1.AND.J.LT.L)E(J)=E(J)+B(J)-R(J)
          E(J)=E(J)/D(J)
          F(J)=0.
          IF(J.GT.1.AND.K.LT.N)F(J)=C(J)-G(J)
          IF(K.LT.N)F(J)=F(J)+P(J)
          F(J)=F(J)/D(J)
  700 CONTINUE
C
C  Solution with lower triangular matrix.
C
C
      DO 850 J=1,L
        IF(J.GE.2)RH(J)=RH(J)-C(J)*VH(J-1)
        IF(J.GE.M+1)RH(J)=RH(J)-B(J)*VH(J-M)
  850   VH(J)=RH(J)/D(J)
C
C  Solution with upper triangular matrix; DE is the difference between
C  new and old solution.
C
      DO 900 J=L,1,-1
        IF(J.LT.L)VH(J)=VH(J)-E(J)*DE(J+1)
        IF(J.LE.L-M)VH(J)=VH(J)-F(J)*DE(J+M)
  900 DE(J)=VH(J)
C
C  Determination of the new solution PSI and the maximum difference
C  between new and old solution.
C
      REST=0.
      DO 950 J=1,L
        PSI(J)=PSI(J)+DE(J)
        IF(ABS(DE(J)).GT.REST)REST=ABS(DE(J))
  950 CONTINUE
      IF(REST.LT.EPS)GO TO 1000
      GO TO 50
 1000 CONTINUE
      TELPSI=TELPSI+NT
      RETURN
      END
```

WEG/WAM/STRIP/HO/STRIP/LIBRARY
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 13:23:50.

```
      FILE  6(KIND='REMOTE')
      FILE  8(KIND='PRINTER')
      BLOCK GLOBALS
        EXPORT(KOEFAF,KOEFRP,BEGWRD)
      END
      CALL FREEZE('TEMPORARY')
      END
C
C
C
      SUBROUTINE KOEFAF(A,B,C,D,E,F,OMO,L,M,N,DT,U,V,X,Y,RHOO,NU,FSTR,
     *                  DEELT,AKERN,TSOURC)
      INTEGER L,M,N,I,J,K,TSOURC
      REAL DEELT,UR,UL,VR,VL,KX,KY,KI,KJ,S1,S2,AKERN,
     *     A(L),B(L),C(L),D(L),E(L),F(L),OMO(L),U(L),V(L),X(L),
     *     Y(L),DT,RHOO,NU,FSTR
C
C
C  Calculation of the value of the coefficients in the discretised
C  vorticity diffusion equation.
C  Boundary condition: the vorticity at the boundaries is zero.
C  The convective term is discretised according to Roache's second
C  upwind differencing scheme.
C
C
      DO 100 J=2,N-1
        DO 100 I=2,M-1
```

```
                  K=I+(J-1)*M
                  UR=(U(K+1)+U(K))/2.
                  UL=(U(K)+U(K-1))/2.
                  VR=(V(K+M)+V(K))/2.
                  VL=(V(K)+V(K-M))/2.
                  KX=(X(K+1)-X(K-1))/2.
                  IF(UR.GE.0..AND.UL.GE.0.)KX=X(K)-X(K-1)
                  IF(UR.LT.0..AND.UL.LT.0.)KX=X(K+1)-X(K)
                  KY=(Y(K+M)-Y(K-M))/2.
                  IF(VR.GE.0..AND.VL.GE.0.)KY=Y(K)-Y(K-M)
                  IF(VR.LT.0..AND.VL.LT.0.)KY=Y(K+M)-Y(K)
                  KI=X(K)-X(K-1)
                  KJ=Y(K)-Y(K-M)
                  S1=(X(K+1)-X(K))/KI
                  S2=(Y(K+M)-Y(K))/KJ
                  IF(J.EQ.2)THEN
                    A(K)=0.
                  ELSE
                    A(K)=(2*NU*DT)/((1+S2)*(KJ**2))
                    IF(VL.GE.0.)A(K)=A(K)+(VL*DT)/KY
                  END IF
                  IF(I.EQ.2)THEN
                    B(K)=0.
                  ELSE
                    B(K)=(2*NU*DT)/((1+S1)*(KI**2))
                    IF(UL.GE.0.)B(K)=B(K)+(UL*DT)/KX
                  END IF
                  C(K)=-1.-(2*NU*DT)*(1./(S1*(KI**2))+1./(S2*(KJ**2)))
                  IF(UR.GE.0.)C(K)=C(K)-(UR*DT)/KX
                  IF(UL.LT.0.)C(K)=C(K)+(UL*DT)/KX
                  IF(VR.GE.0.)C(K)=C(K)-(VR*DT)/KY
                  IF(VL.LT.0.)C(K)=C(K)+(VL*DT)/KY
                  IF(I.EQ.M-1)THEN
                    D(K)=0.
                  ELSE
                    D(K)=(2*NU*DT)/(S1*(1+S1)*(KI**2))
                    IF(UR.LT.0.)D(K)=D(K)-(UR*DT)/KX
                  END IF
                  IF(J.EQ.N-1)THEN
                    E(K)=0.
                  ELSE
                    E(K)=(2*NU*DT)/(S2*(1+S2)*(KJ**2))
                    IF(VR.LT.0.)E(K)=E(K)-(VR*DT)/KY
                  END IF
C
C  QF and QG give the strength of the source at the edge of the
C  actuator strip as a function of time and place.
C
                  IF(TSOURC.EQ.1)F(K)=-OMO(K)-DT*QF(RHOO,FSTR,X(K),Y(K),DEELT,
     *                                                   AKERN)
                  IF(TSOURC.EQ.2)F(K)=-OMO(K)-DT*QG(RHOO,FSTR,X(K),Y(K),DEELT,
     *                                                   AKERN)
100   CONTINUE
        DO 200 J=1,N,N-1
          DO 200 I=1,M
            K=I+(J-1)*M
            A(K)=0.
            B(K)=0.
            C(K)=1.
            D(K)=0.
            E(K)=0.
200         F(K)=0.
        DO 300 I=1,M,M-1
          DO 300 J=1,N
            K=I+(J-1)*M
            A(K)=0.
            B(K)=0.
            C(K)=1.
            D(K)=0.
            E(K)=0.
300         F(K)=0.
        RETURN
        END
C
C
C
        SUBROUTINE KOEFRP(X,Y,L,M,N,R,G,H,KA,P,Q,OMO)
        INTEGER K,I,J,L,M,N
        REAL KI,KJ,S1,S2,X(L),Y(L),R(L),G(L),H(L),KA(L),P(L),Q(L),OMO(L)
C
C  Determination of the coefficients in the discretised Poisson PSI
C  equation to calculate the streamfunction.
C  Boundary condition used here: on the boundaries, the streamfunction
C  and its first derivative are zero.
C
        DO 100 J=3,N-2
          DO 100 I=3,M-2
            K=I+(J-1)*M
              KJ=Y(K)-Y(K-M)
              S2=(Y(K+M)-Y(K))/KJ
              R(K)=2./((1.+S2)*(KJ**2))
              P(K)=R(K)/S2
              KI=X(K)-X(K-1)
              S1=(X(K+1)-X(K))/KI
              G(K)=2./((1.+S1)*(KI**2))
              KA(K)=G(K)/S1
              H(K)=-2./(S2*(KJ**2))-2./(S1*(KI**2))
              Q(K)=-OMO(K)
100   CONTINUE
        DO 200 J=1,N
```

```
          DO 200 I=1,M
            K=I+(J-1)*M
            IF(I.EQ.2.OR.I.EQ.M-1.OR.J.EQ.2.OR.J.EQ.N-1)THEN
              R(K)=0.
              G(K)=0.
              H(K)=1.
              KA(K)=0.
              P(K)=0.
              Q(K)=0.
            END IF
            IF(I.EQ.1.OR.I.EQ.M.OR.J.EQ.1.OR.J.EQ.N)THEN
              P(K)=0.
              R(K)=0.
              G(K)=0.
              KA(K)=0.
              H(K)=1.
              Q(K)=0.
            END IF
  200   CONTINUE
        RETURN
        END
C
C
        REAL FUNCTION QF(RHOO,FSTR,XI,YI,DEELT,AKERN)
        REAL RHOO,FSTR,DEELT,XI,YI,H,K
C
C  This function gives the strenght of the cosh**2 source.
C
        K=4.14677/AKERN
        IF(ABS(K*XI).GT.10..OR.ABS(K*YI).GT.10.)THEN
        QF=0.
        ELSE
        H=((K**2)/(4.*RHOO))/((COSH(K*XI))**2)
        QF=-H*FSTR*DEELT/((COSH(K*YI))**2)
        END IF
        RETURN
        END
C
C
C
        REAL FUNCTION QG(RHOO,FSTR,A,B,DEELT,AKERN)
        REAL A,B,DEELT,AKERN,RHOO,FSTR
C
C This function gives the strength of the source, proposed by Gijs van
C Kuik.
C
        IF(ABS(A).LT.16.9.AND.ABS(B).LT.AKERN.AND.ABS(B).GT.1.E-15)THEN
        QG=-2.3*FSTR*DEELT*(B**2)/(RHOO*ABS(B)*((COSH(4.6*A)*AKERN)**2))
        ELSE
          QG=0.
        END IF
        END
```

```
        SUBROUTINE BEGWRD(L,M,N,U,V,OMO,OMN,PSI,DT,FSTR)
        INTEGER I,J,K,L,M,N
        REAL C,A,B,U(L),V(L),OMO(L),OMN(L),PSI(L)
C
C Here, the vorticity, streamfunction and velocities obtain starting
C values.
C
        DO 10 I=1,L
          OMO(I)=0.
          OMN(I)=0.
   10     PSI(I)=0.
        DO 20 J=1,N
          B=2.*(J-1)/(N-1)-1.
          DO 20 I=1,M
            C=500.*DT*FSTR
            A=2.*(I-1)/(M-1)-1.
            K=I+(J-1)*M
            IF(B**2.GT..0001)U(K)=C*B*(1.-B**2)/SQRT(B**2)*(1.-A**2)
            IF(B**2.LE..0001)U(K)=C*99.99*B*(1.-A**2)
            IF(A**2.GT..0001)V(K)=-C*A*(1.-A**2)/SQRT(A**2)*(1.-B**2)
   20       IF(A**2.LE..0001)V(K)=-C*99.99*A*(1-B**2)
        RETURN
        END
```

```
STRIP/DEVELOP
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:13:32.

C  Strip/develop calculates the time-dependent velocity field,
C  generated by a finite actuator strip.
C  The theory behind this program is discussed in the report R-898-A
C  of the faculty of Physics, Eindhoven Technical University.
C
FILE  8(KIND='PRINTER')
FILE  5(KIND='REMOTE')
      BLOCK GLOBALS
$INCLUDE'PLOTTER/F77/DECLARATION ON APPL.'
C
C  LIB contains the subroutine MARQUA,
C  MLIB contains the subroutine MSTIMP.
C
        LIBRARY LIB(TITLE='(RCLC1)OBJECT/FSLIB/CP ON USER1.')
        LIBRARY NAGF(TITLE='LIBRARY/NAGFLIB/F ON APPL.')
        LIBRARY MLIB(TITLE='OBJECT/STRIP/MAINBIEB.')
      END
$INCLUDE'NAGFLIB/F/FORTRAN/F04ATF ON APPL'
$INCLUDE'PLOTTER/F77/SKIPP ON APPL'
$INCLUDE'PLOTTER/F77/JOINOB ON APPL'
$INCLUDE'PLOTTER/F77/SURF2 ON APPL'
$INCLUDE'PLOTTER/F77/DISPOB ON APPL'
$INCLUDE'PLOTTER/F77/FTEXT ON APPL'
$INCLUDE'PLOTTER/F77/MAPOB1 ON APPL'
$INCLUDE'PLOTTER/F77/CLEARO ON APPL'
$INCLUDE'PLOTTER/F77/DRAWOB ON APPL'
$INCLUDE'PLOTTER/F77/FAXIS ON APPL'
$INCLUDE'PLOTTER/F77/POLYGN ON APPL'
$INCLUDE'PLOTTER/F77/AXISCO ON APPL'
$INCLUDE'PLOTTER/F77/NEWOBJ ON APPL'
C
C  In the following, the text of the various subroutines is included
C  in the text of this programme.
C  This is done in this way, because in most of the subroutines the
C  same variables are used as in the main programme by use of COMMON-
C  blocks. This reduces the computing time.
C
$INCLUDE'STRIP/BIEB/DERIVA'
$INCLUDE'STRIP/BIEB/WERVLY'
$INCLUDE'STRIP/BIEB/MARQUA'
$INCLUDE'STRIP/BIEB/STAROP'
$INCLUDE'STRIP/BIEB/STAREX'
$INCLUDE'STRIP/BIEB/SETCOE'
$INCLUDE'STRIP/BIEB/TRANIO'
$INCLUDE'STRIP/BIEB/KOWDV'
$INCLUDE'STRIP/BIEB/RVWOME'
$INCLUDE'STRIP/BIEB/MSTIMP'
$INCLUDE'STRIP/BIEB/XYTOOM'
$INCLUDE'STRIP/BIEB/FIE'
$INCLUDE'STRIP/BIEB/TRANIP'
$INCLUDE'STRIP/BIEB/KOPOIS'
$INCLUDE'STRIP/BIEB/RVWPSI'
$INCLUDE'STRIP/BIEB/TRVELO'
$INCLUDE'STRIP/BIEB/REGGRI'
$INCLUDE'STRIP/BIEB/DRWGRD'
$INCLUDE'STRIP/BIEB/DRAWUV'
$INCLUDE'STRIP/BIEB/DRAWOM'
$INCLUDE'STRIP/BIEB/UITVOE'
      LOGICAL KOPJE,LDGRID,LDOME,LDVEL,LSTART,LCOME,LCGRID,LCPUV,
     *        LOFILE,LOUTPU
      CHARACTER*30 NIFGRI,NIFRES,NOFGRI,NOFRES
      INTEGER FILENO,NCONLN,NDUM,NITMO,NITMP,NITTO,NITTP
      REAL XWMIN,XWMAX,YWMIN,YWMAX,DUM
$INCLUDE'STRIP/BIEB/COMMON'
C
C  STRIP/BIEB/COMMON contains the declaration of the variables and
C  the common blocks.
C
C  STRIP/PARAM contains the parameters that direct the calculations:
C  XMIN: x-coordinate of left boundary
C  XMAX: "     "      "  right   "
C  YMAX: y-coordinate of upper boundary
C  D: y-coordinate of upper edge of actuator strip.
C  DAKS: ETA-coordinate of upper edge.
C  M: number of grid points in x-direction.
C  N: number of grid points in y-direction.
C  TBEGIN: starting time of the calculation.
C  TEIND: ending time.
C  NT: number of time steps.
C  EPOMME: mean accuracy in the calculation of the vorticity
C  EPOMMA: maximum   "    "  "      "      "  "      "
C  EPSIME: mean accuracy "  "      "      "  "  streamfunction
C  EPSIMA: maximum   "    "  "      "      "  "      "
C  NITMO: maximum number of iterations in calculation of vorticity
C  NITMP:    "      "    "     "       "     "     "  streamfunction
C  XTIMES: ratio between mean vorticity and vorticity of vortex sheet
C  STOPCR: stopcriterion in the determination of least-squares fit to
C          the points with local maximum vorticity.
C  LSTART: logical: if true, a starting  grid is computed and no input
C          file is read from disc.
C  LCOME: logical: if true, the vorticity is computed.
C  LCGRID:    "      "   "    a new  grid is set up.
C  LCPUV:     "      "   "   streamfunction and velocity are calculated.
C  LOFILE:    "      "   "   the results are saved on disc.
C  LOUTPU:    "      "   "   the results are printed.
C  KOPJE :    "      "   "   the calculation parameters are printed if
C         the results are printed.
C  NIFGRI,NIFRES,NOFGRI,NOFRES: Names of Input or Output Files for the
```

```
C                                       GRId and partial derivatives or for the
C                                       RESults.
C
      OPEN(10,FILE='STRIP/PARAM.',STATUS='OLD')
      READ(10,*)XMIN,XMAX,YMAX,D,DAKS,M,N,NHETA,TBEGIN,TEIND,NT
      READ(10,*)EPOMME,EPOMMA,EPSIME,EPSIMA,NITMO,NITMP
      READ(10,*)XTIMES,STOPCR,NOUTQF,EA,ER,ALFAO,ALFAP,NPAR,WIDTH
      READ(10,*)LDGRID,LDOME,LDVEL,NCONLN,DIAGNO
      READ(10,*)XIXMIN,XIXMAX,ETYMIN,ETYMAX,NCPL,NITPMX,NITPMY
      READ(10,*)XWMIN,XWMAX,YWMIN,YWMAX
      READ(10,*)LSTART,LCOME,LCGRID,LCPUV,LOFILE,LOUTPU,KOPJE
      READ(10,*)NIFGRI
      READ(10,*)NIFRES
      READ(10,*)NOFGRI
      READ(10,*)NOFRES
      L=M*N
C
C  STAROP sets up a physical grid plus a computational grid and sets
C  OM,PSI,U and V zero.
C
C  STAREX only sets up a computational grid.
C
      IF(LSTART)CALL STAROP
      IF(.NOT.LSTART)CALL STAREX
      DT=(TEIND-TBEGIN)/NT
      FILENO=8
      RHOO=1.2
      NU=1.45E-5
      IF(LSTART)GO TO 70
      OPEN(20,STATUS='OLD',FILE=NIFGRI)
      OPEN(30,STATUS='OLD',FILE=NIFRES)
      READ(20,*)DUM,DUM,DUM,DUM,DUM,NDUM,NDUM,DUM,DUM,NDUM
      READ(20,*)DUM,DUM,DUM,DUM
      READ(20,*)DUM,DUM,DUM,DUM,DUM,DUM,NDUM,DUM
      READ(30,*)DUM,DUM,DUM,DUM,DUM,NDUM,NDUM,DUM,DUM,NDUM
      READ(30,*)DUM,DUM,DUM,DUM
      READ(30,*)DUM,DUM,DUM,DUM,DUM,DUM,NDUM,DUM
      DO 50 J=0,N-1
      READ(20,*)(X(J*M+I),Y(J*M+I),KSIX(J*M+I),KSIXX(J*M+I),EX(J*M+I),
     *  EY(J*M+I),EXX(J*M+I),EYY(J*M+I),I=1,M)
      READ(30,*)(OM(J*M+I),PSI(J*M+I),U(J*M+I),V(J*M+I),I=1,M)
   50 CONTINUE
      CLOSE(20,STATUS='CRUNCH')
      CLOSE(30,STATUS='CRUNCH')
   70 CONTINUE
C
C  Beginning of the loop, that is repeated every time step.
C
      TIT=0
  100 CONTINUE
      TIT=TIT+1
      IF(.NOT.LCOME)GO TO 107
C
C  SETCOE calculates the coefficients for the vorticity diffusion
C        equation.
C  TRANIO transforms this coefficients into the computational plane.
C  KOWDV determines the coefficients for the discretised, transformed
C        vorticity diffusion equation.
C  RVWOME determines boundary conditions.
C
      CALL SETCOE
      CALL TRANIO
      CALL KOWDV
      CALL RVWOME
      NITO=0
C
C  MSTIMP caculates a solution to a nine point molecule difference
C        scheme, according to the modified strongly implicit procedure
C
      CALL MSTIMP(M,N,OM,AO,BO,CO,DO,EO,FO,GO,HO,IO,JO,H1,H2,H3,H4,H5,
     *     H6,H7,H8,H9,H10,H11,H12,ALFAO,EPOMME,EPOMMA,NITMO,NITO)
      NITTO=NITTO+NITO
      WRITE(5,105)TIT,NITO
  105 FORMAT(1X,'Time iteration number ',I4,/,1X,
     *'Vorticity diffusion equation solved,'I4,' iterations used.')
  107 CONTINUE
      IF(.NOT.LCGRID)GO TO 110
C
C  WERVLY computes the coordinates of the local maxima in the vorticity
C        distribution and fits a function to these points.
C        This function then serves as a base for the transformation of
C        the grid.
C  XYTOOM generates a grid by calculation of the transformation from
C        physical to computational grid.
C
      CALL WERVLY(XTIMES,STOPCR,NOUTQF,EA,ER)
      CALL XYTOOM
      WRITE(5,120)TIT
  120 FORMAT(1X,'Time iteration number ',I4,/,1X,'New grid plus ',
     * 'partial derivatives determined.')
  110 CONTINUE
      IF(.NOT.LCPUV)GO TO 115
      DO 112 I=1,L
  112 ZP1(I)=-OM(I)
C
C  TRANIP, KOPOIS and RVWPSI: same as TRANIO, KOWDV and RVWOME,
C  respectively, but working on the Poisson-PSI equation.
C
      CALL TRANIP
      CALL KOPOIS
```

```
      CALL RVWPSI
      NITP=0
      CALL MSTIMP(M,N,PSI,AP,BP,CP,DP,EP,FP,GP,HP,IP,JP,H1,H2,H3,H4,H5,
     *     H6,H7,H8,H9,H10,H11,H12,ALFAP,EPSIME,EPSIMA,NITMP,NITP)
      NITTP=NITTP+NITP
C
C  TRVELO computes the transformed velocity components from the
C         streamfunction.
C
      CALL TRVELO
      WRITE(5,114)TIT,NITP
  114 FORMAT(1X,'Time iteration number',I4,/,1X,
     *'streamfunction calculated: ',I4,' iterations used.')
  115 CONTINUE
      IF(TIT.LT.NT)THEN
        DO 140 I=1,L
  140   OMO(I)=OM(I)
      END IF
      IF(TIT.LT.NT)GO TO 100
  150 CONTINUE
      IF(.NOT.LOFILE)GO TO 205
      OPEN(20,STATUS='NEW',FILE=NOFGRI)
      OPEN(30,STATUS='NEW',FILE=NOFRES)
C
C  In FILE20, the coordinates of the gridpoints and the partial
C  derivatives of the transformation are stored.
C  FILE30 is filled with the vorticity, streamfunction and velocity
C  components in each grid point.
C
      WRITE(20,*)XMIN,XMAX,YMAX,D,DAKS,M,N,TBEGIN,TEIND,NT
      WRITE(30,*)XMIN,XMAX,YMAX,D,DAKS,M,N,TBEGIN,TEIND,NT
      WRITE(20,*)EPOMME,EPOMMA,EPSIME,EPSIMA
      WRITE(30,*)EPOMME,EPOMMA,EPSIME,EPSIMA
      WRITE(20,*)XTIMES,STOPCR,EA,ER,ALFAO,ALFAP,NPAR,WIDTH
      WRITE(30,*)XTIMES,STOPCR,EA,ER,ALFAO,ALFAP,NPAR,WIDTH
      DO 200 J=0,N-1
      WRITE(20,*)(X(J*M+I),Y(J*M+I),KSIX(J*M+I),KSIXX(J*M+I),EX(J*M+I),
     *  EY(J*M+I),EXX(J*M+I),EYY(J*M+I),I=1,M)
      WRITE(30,*)(OM(J*M+I),PSI(J*M+I),U(J*M+I),V(J*M+I),I=1,M)
  200 CONTINUE
      CLOSE(20,STATUS='CRUNCH')
      CLOSE(30,STATUS='CRUNCH')
  205 CONTINUE
      IF(.NOT.LDGRID)GO TO 210
C
C  DRWGRD makes a plot of the physical grid with lines KSI=constant and
C           ETA=constant.
C
       CALL DRWGRD(X,XWMIN,XWMAX,M,Y,YWMIN,YWMAX,N,'X-axis   ',
     *  ' Y-axis ',NOFGRI)
  210 CONTINUE
      IF(.NOT.LDVEL)GO TO 220
C
C  DRAWUV makes a plot of streamlines (PSI=constant); the number of
C         streamlines drawn is NCONLN.
C
      CALL DRAWUV(X,Y,PSI,M,N,XWMIN,XWMAX,YWMIN,YWMAX,TBEGIN,TEIND,
     *             NT,NCONLN)
  220 CONTINUE
      IF(.NOT.LDOME)GO TO 230
C
C  REGGRI,SHIFTO and DRAWOM produce a plot of the vorticity in a
C                           perspective view.
C
      CALL REGGRI
      CALL SHFTOY
      CALL DRAWOM(X,Y,M,N,OM,XWMIN,XWMAX,YWMIN,YWMAX,NOFGRI)
  230 CONTINUE
      IF(.NOT.LOUTPU)GO TO 300
C
C  UITVOE prints the results, and, if KOPJE is true, also a
C         list of the calculation parameters.
C
      CALL UITVOE(KOPJE,L,X,Y,OM,PSI,U,V,FILENO)
  300 CONTINUE
 1000 CONTINUE
      END
```

STRIP/PARAM
DATE & TIME PRINTED: MONDAY, FEBRUARY 29, 1988 @ 13:28:37.

```
-100.,100.,100.,5.,50.,41,21,0.000,0.001,1
 1.E-5,1.E-5,1.E-5,1.E-5,200,200
 2.0,12,1,1.E-3,1.E-2,.5,.5,5,.25
F,F,F,10
1.E-1,10.,1.E-1,10.,5,500,1000,6
-20.,20.,0.,20.
F,T,T,T,T,T,T
'AS/GRID/41B21/1/1000.'
'AS/OPUV/41B21/1/1000.'
'AS/GRID/41B21/1/1000.'
'AS/OPUV/41B21/1/1000.'
'DATA FOR THE CALCULATION OF FLOW THROUGH FINITE ACTUATOR STRIP'
'LINE 1:  XMIN,XMAX,YMAX,D,DAKS,M,N,TBEGIN,TEIND,NT'
'LINE 2:  EPOMME,EPOMMA,EPSIME,EPSIMA,NITMO,NITMP'
'LINE 3:  XTIMES,STOPCR,NOUTQF,EA,ER,ALFAO,ALFAP,NPAR,WIDTH'
```

```
     'LINE 4: LDGRID,LDOME,LDVEL,NCONLN'
     'LINE 5: XIXMIN,XIXMAX,ETYMIN,ETYMAX,NCPL,NITPMX,NITPMY,NHETA'
     'LINE 6: XWMIN,XWMAX,YWMIN,YWMAX'
     'LINE 7: LSTART,LCOME,LCGRID,LCPUV,LOFILE,LOUTPU,KOPJE'
     'LINE 8: NIFGRI'
     'LINE 9: NIFRES'
     'LINE 10: NOFGRI'
     'LINE 11: NOFRES'
```

STRIP/BIEB/MARQUA
DATE & TIME PRINTED: FRIDAY, DECEMBER 4, 1987 @ 15:49:31.

```
      SUBROUTINE MARQUA(N,R,M,P,RES,STOPCR,EA,ER,ITMAX,OUTPUT,FNR)
      INTEGER N,M,STOPCR,ITMAX,OUTPUT,FNR
      REAL EA,ER,R(N),P(M)
      EXTERNAL RES
      IN LIBRARY LIB
      END
```

STRIP/BIEB/STAROP
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:15:00.

```
      SUBROUTINE STAROP
C
C This routine sets up a starting grid.
C The lines ksi=constant and eta=constant are straight lines.
C In x-direction the grid points are distributed in the following way:
C m-4 grid points are equally distributed between x=xmin/3 and
C x=xmax/3.
C The outer 4 points have x-coordinates xmin,xmin/2,xmax/2 and xmax.
C
C In y-direction: between y=0 and y=3*D 3*NY3 points are equally
C distributed. The other points have coordinates ymax, ymax/2,
C ymax/3, etcetera.
C
C The partial derivatives ksix, ksixx, ey and eyy are determined from
C a second order Taylor series expansion. ksixx and eyy are set zero
C on the boundaries.
C
C Finally, the starting values of PSI, OM, U, V and EXY are set zero.
C
      REAL S1,S2,S22
$INCLUDE'STRIP/BIEB/COMMON'
      MH=(M+1)/2
      NH=(N+1)/2
      NY3=3*(NHETA-1)+1
      NX3=M-6
      DEX=(XMAX-XMIN)/(3.*FLOAT(NX3+1))
      DEY=3*D/FLOAT(NY3-1)
      DEXI=(XMAX-XMIN)/(M-1)
      DEET=YMAX/(N-1)
      DO 50 J=1,N
        DO 50 I=1,M
          K=(J-1)*M+I
          KSI(K)=XMIN+(I-1)*DEXI
          ETA(K)=(J-1)*DEET
   50 CONTINUE
      DO 55 I=4,MH-1
   55   X(I)=XMIN/3+(I-3)*DEX
      DO 57 I=MH+1,M-3
   57   X(I)=XMAX/3-(NX3-I+4)*DEX
      X(1)=XMIN
      X(2)=XMIN/2
      X(3)=XMIN/3
      X(MH)=0.
      X(M-2)=XMAX/3
      X(M-1)=XMAX/2
      X(M)=XMAX
      DO 60 J=2,NY3
   60   Y((J-1)*M+1)=(J-1)*DEY
      Y(1)=0.
      DO 65 I=1,N-NY3
   65 Y((N-I)*M+1)=YMAX/I
      DO 70 I=2,M-1
        S1=X(I)-X(I-1)
        S2=(X(I+1)-X(I))/S1
        S22=S2*S2
   70   KSIX(I)=(KSI(I+1)-S22*KSI(I-1)+(S22-1)*KSI(I))/(S1*S2*(1+S2))
      KSIX(1)=KSIX(2)+(X(1)-X(2))*(KSIX(3)-KSIX(2))/(X(3)-X(2))
      KSIX(M)=KSIX(M-1)+(X(M)-X(M-1))*(KSIX(M-1)-KSIX(M-2))/
     *          (X(M-1)-X(M-2))
      DO 80 J=2,N-1
        K=(J-1)*M+1
        S1=Y(K)-Y(K-M)
        S2=(Y(K+M)-Y(K))/S1
   80   EY(K)=(ETA(K+M)-S22*ETA(K-M)+(S22-1)*ETA(K))/(S1*S2*(1+S2))
      EY(1)=EY(M+1)+(Y(1)-Y(M+1))*(EY(1+2*M)-EY(1+M))/(Y(1+2*M)-Y(1+M))
      K=(N-1)*M+1
      EY(K)=EY(K-M)+(Y(K)-Y(K-M))*(EY(K-M)-EY(K-2*M))/(Y(K-M)-Y(K-2*M))
      DO 85 I=2,M-1
        S1=X(I)-X(I-1)
        S2=(X(I+1)-X(I))/S1
        S22=S2*S2
   85 KSIXX(I)=(KSIX(I+1)-S22*KSIX(I-1)+(S22-1)*KSIX(I))/(S1*S2*(1+S2))
      KSIXX(1)=0.
      KSIXX(M)=0.
      DO 90 J=2,N-1
        K=(J-1)*M+1
        S1=Y(K)-Y(K-M)
        S2=(Y(K+M)-Y(K))/S1
```

```
   90  EYY(K)=(EY(K+M)-S22*EY(K-M)+(S22-1)*EY(K))/(S1*S2*(1+S2))
       EYY(1)=0.
       EYY((N-1)*M+1)=0.
       DO 170 J=1,N
         DO 170 I=1,M
           K=(J-1)*M+I
           X(K)=X(I)
           Y(K)=Y((J-1)*M+1)
           KSIX(K)=KSIX(I)
           KSIXX(K)=KSIXX(I)
           EY(K)=EY((J-1)*M+1)
           EYY(K)=EYY((J-1)*M+1)
  170      CONTINUE
       DO 100 I=1,L
         EXY(I)=0.
         OMO(I)=0.
         PSI(I)=0.
         U(I)=0.
  100    V(I)=0.
       END
```

STRIP/BIEB/STAREX
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:16:50.

```
       SUBROUTINE STAREX
C
C  setting values of the eta- and ksi-coordinates of the computational
C  grid.
C
       INTEGER L,I,M,N,J,K
       REAL XMIN,XMAX,YMAX,
     *      DEET,DEXI,KSI(10000),ETA(10000)
       COMMON /SIZE/M,N,L,/CONFIG/XMIN,XMAX,YMAX
       COMMON /TRCOOR/KSI,ETA
       DEXI=(XMAX-XMIN)/(M-1)
       DEET=YMAX/(N-1)
       DO 50 J=1,N
         DO 50 I=1,M
           K=(J-1)*M+I
           KSI(K)=XMIN+(I-1)*DEXI
           ETA(K)=(J-1)*DEET
   50  CONTINUE
       END
```

STRIP/BIEB/SETCOE
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:23:18.

```
C
C  In this subroutine, the coefficients for the ordinary vorticity
C  diffusion equation are given a value: in grid point I the equation
C  reads:
C
C  dOM/dt = Z01(I) + Z02(I)dOM/dx + Z03(I)dOM/dy + Z04(I)dOMdOM/dxdx +
C           + Z05(I)dOMdOM/dydy
C
C  Om is the vorticity.
      SUBROUTINE SETCOE
$INCLUDE'STRIP/BIEB/COMMON'
      DO 100 I=1,L
        Z01(I)=(-1./RHOO)*SOURCE(X(I),Y(I))
        Z02(I)=-U(I)
        Z03(I)=-V(I)
        Z04(I)=NU
        Z05(I)=NU
  100 CONTINUE
      END
C**********************************************************************
C
      REAL FUNCTION SOURCE(X,Y)
      REAL X,Y,W,D,K
C
C  The strength of the vorticity-source at X,Y is given by this
C  function. W is the width of the point-like source.
C
C  Strength of the source at |X|=W  or |Y-D|=W is 1/1000 of the maximum
C
      COMMON /DIAM/D,W
      K=4.14677/W
      IF(ABS(K*X).GT.10..OR.ABS(K*(Y-D)).GT.10.)THEN
        SOURCE=0.
      ELSE
        SOURCE=(-K**2)/(4.*(COSH(K*X)*COSH(K*(Y-D)))**2)
      END IF
      END
```

STRIP/BIEB/TRANIO
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:25:50.

```
      SUBROUTINE TRANIO
$INCLUDE'STRIP/BIEB/COMMON'
C
C transformation of the coefficients in the vorticity-diffusion equation
C  into the computational plane.
C
      DO 100 I=1,L
        ZOT1(I)=Z01(I)
        ZOT2(I)=Z02(I)*KSIX(I)+Z04(I)*KSIXX(I)
        ZOT3(I)=Z02(I)*EX(I)+Z03(I)*EY(I)+Z04(I)*EXX(I)+Z05(I)*EYY(I)
        ZOT4(I)=Z04(I)*(KSIX(I)**2)
        ZOT5(I)=Z04(I)*(EX(I)**2)+Z05(I)*(EY(I)**2)
        ZOT6(I)=2*Z04(I)*KSIX(I)*EX(I)
  100 CONTINUE
      END
```

STRIP/BIEB/KOWDV
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:27:12.

```
      SUBROUTINE KOWDV
C
C  determination of the coefficients for the discretised, transformed
C  vorticity diffusion equation.
C
      REAL DEXI,DEET
$INCLUDE'STRIP/BIEB/COMMON'
      DEXI=(XMAX-XMIN)/(M-1)
      DEET=YMAX/(N-1)
      DO 100 I=1,L
        AO(I)=DT*ZOT6(I)/(4*DEXI*DEET)
        BO(I)=DT*(-ZOT3(I)/(2*DEET)+ZOT5(I)/(DEET**2))
        CO(I)=-AO(I)
        DO(I)=DT*(-ZOT2(I)/(2*DEXI)+ZOT4(I)/(DEXI**2))
        EO(I)=-1.+DT*(-2*ZOT4(I)/(DEXI**2)-2*ZOT5(I)/(DEET**2))
        FO(I)=DT*(ZOT2(I)/(2*DEXI)+ZOT4(I)/(DEXI**2))
        GO(I)=-AO(I)
        HO(I)=DT*(ZOT3(I)/(2*DEET)+ZOT5(I)/(DEET**2))
        IO(I)=AO(I)
        JO(I)=-OMO(I)-DT*ZOT1(I)
  100 CONTINUE
      END
C
```

STRIP/BIEB/RVWOME
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:30:52.

```
      SUBROUTINE RVWOME
C
C  this subroutine calculates the effect of the boundary conditions on
C  the coefficients in the differential scheme of the vorticity-diffu-
C  sion equation.
C
      REAL DEXI,DEET,Q(10000),A
$INCLUDE'STRIP/BIEB/COMMON'
      DEXI=(XMAX-XMIN)/(M-1)
      DEET=YMAX/(N-1)
C
C  boundary conditions for the lower boundary y=0: mirror
C
      DO 100 I=1,M
        GO(I)=GO(I)-AO(I)
        HO(I)=HO(I)-BO(I)
        IO(I)=IO(I)-CO(I)
  100 CONTINUE
C
C right boundary: X=XMAX, vorticity equal to zero.
C
      DO 300 J=1,N
        K=J*M
        EO(K)=1.
        AO(K)=0.
        BO(K)=0.
        CO(K)=0.
        DO(K)=0.
        FO(K)=0.
        JO(K)=0.
  300 CONTINUE
C
C upper boundary: Y=YMAX, vorticity equal zero
C
      DO 400 I=M,1,-1
        K=I+(N-1)*M
        EO(K)=1.
        AO(K)=0.
        BO(K)=0.
        CO(K)=0.
        DO(K)=0.
        FO(K)=0.
        JO(K)=0.
  400 CONTINUE
C
C  left boundary: X=XMIN, vorticity equal zero
C
      DO 500 J=N,1,-1
        K=(J-1)*M+1
        EO(K)=1.
        IO(K)=0.
        FO(K)=0.
        CO(K)=0.
        HO(K)=0.
        BO(K)=0.
        JO(K)=0.
C
C  setting coefficients for points outside grid zero
C
        AO(K)=0.
        DO(K)=0.
        GO(K)=0.
  500 CONTINUE
C
      DO 510 I=1,M
        K=I
        AO(K)=0.
        BO(K)=0.
  510   CO(K)=0.
```

```
      DO 520 J=1,N
        K=J*M
        CO(K)=0.
        FO(K)=0.
  520   IO(K)=0.
      DO 530 I=M,1,-1
        K=(N-1)*M+I
        GO(K)=0.
        HO(K)=0.
  530   IO(K)=0.
C
C Vorticity on lower,left and upper boundary equal zero: coefficients
C for these points at rows next to boundaries equal zero.
C
      DO 550 I=1,M
        K=I+M
        AO(K)=0.
        BO(K)=0.
  550   CO(K)=0.
      DO 560 J=1,N
        K=(J-1)*M+2
        AO(K)=0.
        DO(K)=0.
  560   GO(K)=0.
      DO 570 I=1,M
        K=(N-2)*M+I
        GO(K)=0.
        HO(K)=0.
  570   IO(K)=0.
      DO 580 J=1,N
        K=J*M-1
        CO(K)=0.
        FO(K)=0.
        IO(K)=0.
  580 CONTINUE
      END
```

STRIP/BIEB/MSTIMP
DATE & TIME PRINTED: WEDNESDAY, JANUARY 27, 1988 @ 12:08:38.

```
      SUBROUTINE MSTIMP(M,N,OM,AA,BA,CA,DA,EA,FA,GA,HA,IA,JA,A,B,C,
     *         D,E,F,G,H,U,DE,VH,R,ALFA,EPOMME,EPOMMA,NITMAX,NITTOT)
      INTEGER M,N,NITMAX,NITTOT
      REAL OM(M*N),AA(M*N),BA(M*N),CA(M*N),DA(M*N),EA(M*N),FA(M*N),
     *     GA(M*N),HA(M*N),IA(M*N),JA(M*N),A(M*N),B(M*N),C(M*N),D(M*N),
     *     E(M*N),F(M*N),G(M*N),H(M*N),U(M*N),DE(M*N),R(M*N),ALFA,
     *     VH(M*N),EPOMME,EPOMMA
      IN LIBRARY MLIB
      END
```

STRIP/BIEB/WERVLY
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:34:24.

```
      SUBROUTINE WERVLY(XTIMES,STOPCR,NOUTQF,EA,ER)
C
C WERVLY calculates a least-squares fit of a function FIE to the local
C maxima of the vorticity distribution.
C
      EXTERNAL RES
      INTEGER M,N,STOPCR,NPAR,NMAX
      REAL SOM,MAX,R(100),EA,ER,X1,X2,X3,X4,X5,D1,D2,D4,D5,F1,F2,F3,F4,
     *   F5,FAKS3,F2AKS3,A,B,OM(10000),X(10000),PAR(10),XTIMES,XM(100),
     *   YM(100),Y(10000)
      COMMON /XMYM/XM,YM,/FIELD/OM,/COOR/X,Y,/SIZE/M,N,/PARAM/NPAR,PAR
      NMAX=1
      DO 100 I=(M+1)/2,M
        MAX=0.
        SOM=0.
        DO 50 J=1,N
          K=(J-1)*M+I
          SOM=SOM+ABS(OM(K))
          IF(ABS(OM(K)).GE.MAX)THEN
            MAX=ABS(OM(K))
            KMAX=K
          END IF
   50   CONTINUE
        SOM=SOM/N
        IF(MAX.GE.XTIMES*SOM.AND.MAX.GT.1.E-20)THEN
          IF(KMAX.LE.2*M.OR.KMAX.GT.(N-2)*M)THEN
            YM(NMAX)=Y(KMAX)
          ELSE
            X1=Y(KMAX-2*M)
            X2=Y(KMAX-M)
            X3=Y(KMAX)
            X4=Y(KMAX+M)
            X5=Y(KMAX+2*M)
            D1=X1-X3
            D2=X2-X3
            D4=X4-X3
            D5=X5-X3
            F1=OM(KMAX-2*M)
            F2=OM(KMAX-M)
            F3=OM(KMAX)
            F4=OM(KMAX+M)
            F5=OM(KMAX+2*M)
            FAKS3=(F4*D2**2-F2*D4**2-F3*(D2**2-D4**2))/(D2*D4*(D2-D4))
            A=D1*D2*(D2**2-D1**2)
            B=D4*D5*(D5**2-D4**2)
            F2AKS3=B*(F1*D2**3-F2*D1**3)-A*(F4*D5**3-F5*D4**3)-
     *           F3*(B*(D2**3-D1**3)-A*(D5**3-D4**3))
```

```
               YM(NMAX)=X3-FAKS3*(.5*(B*(D2-D1)*(D1*D2)**2-
     *              A*(D5-D4)*(D4*D5)**2))/F2AKS3
             END IF
             XM(NMAX)=X(KMAX)
             NMAX=NMAX+1
           END IF
  100   CONTINUE
        IF(NMAX.GE.2*NPAR)THEN
        NSTOP=STOPCR
C
C  MARQUA calculates the set of parameters that minimises the residual
C         vector res, calculated in the function RES.
C
        CALL MARQUA(NMAX,R,NPAR,PAR,RES,NSTOP,EA,ER,100,NOUTQF,86)
        END IF
        END
C ********************************************************************
C
        SUBROUTINE RES(N,PAR,NMAX,R)
C
C RES calculates the residu between the curve-fit and the measured
C values for the local maxima in the vorticity-distribution.
C
        INTEGER NMAX,I,N
        REAL R(NMAX),PAR(N),XM(100),YM(100),F
        COMMON /XMYM/XM,YM,/PARAM/NPAR,/DIAM/D
        DO 100 I=1,NMAX
          IF(ABS(XM(I)).LT.1.E-2)THEN
            SIGMA=100.
          ELSE
            SIGMA=1./ABS(XM(I))
          END IF
          F=FIE(XM(I),PAR,D)
          R(I)=(F-YM(I))/SIGMA
  100   CONTINUE
        END
STRIP/BIEB/XYTOOM
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:35:57.


      SUBROUTINE XYTOOM
      LOGICAL LFIT
      INTEGER NC,DELER1,DELER2,NPENAL,NITPEN
      REAL A,AOMRO(100),OMT,OMA(10000),HULP(100),DELT,A1,A2,DEY,
     *     XN(10000),YN(10000),DELTA(100),ALF1(100),ALF2(100),HIFX(100)
$INCLUDE'STRIP/BIEB/COMMON'
      DEXI=(XMAX-XMIN)/(M-1)
      DEET=YMAX/(N-1)
C
C  This subroutine constructs a new grid, in which the grid points
C  are distributed proportional to the distribution of vorticity.
C  The two rows of gridpoints at the edges of the domain are held fixed
C
C  for fitting of new x-coordinates first summation of vorticity
C  per column.(X=constant)
C
      MH=(M+1)/2
      DO 20 I=1,M
        AOMRO(I)=0.
        DO 10 J=1,N-1
          K=(J-1)*M+I
          AOMRO(I)=AOMRO(I)+OM(K)*OM(K)
 10     CONTINUE
        AOMRO(I)=SQRT(AOMRO(I))
 20   CONTINUE
C
C  creating new AOMRO(I) in such a way that AOMRO only increases
C  with increasing I.
C
      DO 30 I=2,M-1
        DELTA(I)=AOMRO(I)-AOMRO(I-1)
        IF(DELTA(I).LT.0.)DELTA(I)=-DELTA(I)
 30   CONTINUE
      DO 40 I=2,M-1
 40     AOMRO(I)=AOMRO(I-1)+DELTA(I)
      NITPEN=0
 41   CONTINUE
      IF(NITPEN.GT.NITPMX)GO TO 49
C
C  Scaling of the values of AOMRO with A1 and A2
C
      A1=(-XMIN-DEXI)/(AOMRO(MH)-AOMRO(2))
      A2=(XMAX-DEXI)/(AOMRO(M-1)-AOMRO(MH))
      DO 42 I=3,MH-1
 42     AOMRO(I)=DEXI+A1*(AOMRO(I)-AOMRO(2))
      DO 44 I=MH+1,M-2
 44     AOMRO(I)=-XMIN+A2*(AOMRO(I)-AOMRO(MH))
      AOMRO(MH)=-XMIN
      AOMRO(1)=0.
      AOMRO(2)=DEXI
      AOMRO(M-1)=XMAX-XMIN-DEXI
      AOMRO(M)=XMAX-XMIN
C
C  Checking the differences between consequent AOMRO, minimum value
C  is XIXMIN.
C
      NPENAL=0
      DO 47 I=3,M-1
```

```
          DELTA(I)=AOMRO(I)-AOMRO(I-1)
          DEX=X(I)-X(I-1)
        IF(DELTA(I)/DEX.LT.XIXMIN)THEN
          IF(DELTA(I).LT.1.E-20)THEN
            DELTA(I)=DEX*XIXMIN
          ELSE
            DELTA(I)=DEX*DEX*XIXMIN*XIXMIN/DELTA(I)
          END IF
          NPENAL=NPENAL+1
        END IF
        IF(DELTA(I)/DEX.GT.XIXMAX)THEN
          DELTA(I)=DEX*DEX*XIXMAX*XIXMAX/DELTA(I)
          NPENAL=NPENAL+1
        END IF
   47   CONTINUE
        DO 48 I=3,M-1
   48     AOMRO(I)=AOMRO(I-1)+DELTA(I)
        IF(NPENAL.GT.0)NITPEN=NITPEN+1
        IF(NPENAL.GT.0)GO TO 41
   49   CONTINUE
C
C A1 and A2 are the coupling factors between the vorticity in a row and
C the value of KSI (the vertical grid lines are spread according to the
C sum of vorticity in a row)
C
        A1=(-XMIN-DEXI)/(AOMRO(MH)-AOMRO(2))
        A2=(XMAX-DEXI)/(AOMRO(M-1)-AOMRO(MH))
C
C  calculation of the values of the new KSI belonging to the old grid
C  points
C
        DO 50 I=3,MH-1
          AOMRO(I)=XMIN+DEXI+A1*(AOMRO(I)-AOMRO(2))
   50   CONTINUE
        DO 60 I=MH+1,M-2
          AOMRO(I)=A2*(AOMRO(I)-AOMRO(MH))
   60   CONTINUE
        AOMRO(1)=XMIN
        AOMRO(2)=XMIN+DEXI
        AOMRO(M-1)=XMAX-DEXI
        AOMRO(M)=XMAX
        AOMRO(MH)=0.
C
C  Now determination of X-coordinates of new gridpoints,using linear
C  interpolation of the new KSI-coordinates
C  The new X-coordinates are put in the array XN
C
        XN(1)=XMIN
        XN(2)=XMIN/2.
        XN(MH)=0.
        XN(M-1)=XMAX/2.
        XN(M)=XMAX
        DO 120 I=3,MH-1
          DO 120 J=3,MH
            IF(AOMRO(J-1).LE.KSI(I).AND.AOMRO(J).GT.KSI(I))THEN
              XN(I)=X(J-1)+(KSI(I)-AOMRO(J-1))*(X(J)-X(J-1))/(AOMRO(J)-
     *                AOMRO(J-1))
            END IF
  120   CONTINUE
        DO 130 I=MH+1,M-2
          DO 130 J=MH+1,M-1
            IF(AOMRO(J-1).LE.KSI(I).AND.AOMRO(J).GT.KSI(I))THEN
              XN(I)=X(J-1)+(KSI(I)-AOMRO(J-1))*(X(J)-X(J-1))/(AOMRO(J)-
     *                AOMRO(J-1))
            END IF
  130   CONTINUE
        DO 140 J=2,N
          DO 140 I=1,M
            K=(J-1)*M+I
  140       XN(K)=XN(I)
C
C  Changing old and new X-coordinates, Y-coordinates stay the same
C
        DO 145 J=1,N
          DO 145 I=1,M
            K=(J-1)*M+I
            XO(K)=X(K)
  145       X(K)=XN(K)
C
C  Now the the values of OM are interpolated for the new X-coordinates
C
        CALL SHFTOX
C
C  For the gridpoints with altered X-coordinate,
C  the same procedure is followed for the coupling between vorticity
C  and new Y-coordinates.
C
        DO 65 I=1,M
   65     YM(I)=FIE(X(I),PAR,D)
        DO 80 I=1,M
          OMA(I)=0.
          DO 70 J=2,N-2
            K=(J-1)*M+I
            DELT=ABS(OM(K))-ABS(OM(K-M))
            IF(DELT.LT.0.)DELT=-DELT
   70       OMA(K)=OMA(K-M)+DELT+ETYMIN*.5*(Y(K+M)-Y(K-M))
          K=(N-2)*M+I
          OMA(K)=ABS(OM(K))-ABS(OM(K-M))
          IF(OMA(K).LT.0.)OMA(K)=-OMA(K)
          OMA(K)=OMA(K-M)+OMA(K)+ETYMIN*(Y(K)-Y(K-M))
   80   CONTINUE
```

```
C
C  the values of OMA in NCPL neighbouring columns are coupled and put
C  in the array H1
C
      DO 85 I=1,M
   85   H1(I)=0.
      NITPEN=0
      LFIT=.TRUE.
      DO 90 J=2,N-1
        DO 90 I=1,M
          K=(J-1)*M+I
          H1(K)=OMA(K)
              DO 86 NC=1,NCPL
                IF(I.GT.NC)H1(K)=H1(K)+OMA(K-NC)/((NC+1)*(NC+1))
                IF(I.LT.M+1-NC)H1(K)=H1(K)+OMA(K+NC)/((NC+1)*(NC+1))
   86         CONTINUE
   90 CONTINUE
      IF(NITPEN.GT.NITPMY)LFIT=.FALSE.
C
C Determination of the coupling coefficients per column ALF1(I) and
C ALF2(I)
C
      DO 105 I=1,M
        DO 105 J=2,N-1
          K=(J-1)*M+I
          IF(Y(K-M).LE.YM(I).AND.Y(K).GT.YM(I))THEN
          H1FX(I)=H1(K-M)+(YM(I)-Y(K-M))*(H1(K)-H1(K-M))/(Y(K)-Y(K-M))
          END IF
  105 CONTINUE
      DO 100 I=1,M
        ALF1(I)=((NHETA-1)*DEET)/H1FX(I)
  100   ALF2(I)=((N-NHETA-1)*DEET)/(H1((N-2)*M+I)-H1FX(I))
C
C  Calculation of the new ETA-coordinates of the old grid points.
C
      DO 109 I=1,M
        H1(I)=0.
  109   H1((N-1)*M+I)=YMAX
      NPENAL=0
      DO 110 J=2,N-1
        DO 110 I=1,M
          K=(J-1)*M+I
          IF(H1(K).LE.H1FX(I))THEN
            H1(K)=ALF1(I)*H1(K)
          ELSE
            H1(K)=(NHETA-1)*DEET+ALF2(I)*(H1(K)-H1FX(I))
          END IF
          IF(.NOT.LFIT)GO TO 110
            H2(K)=H1(K)-H1(K-M)
            DEY=Y(K)-Y(K-M)
            IF(H2(K)/DEY.GT.ETYMAX)THEN
              H2(K)=DEY*DEY*ETYMAX*ETYMAX/H2(K)
              NPENAL=NPENAL+1
            END IF
            IF(H2(K)/DEY.LT.ETYMIN)THEN
              IF(H2(K).LT.1.E-20)THEN
                H2(K)=DEY*ETYMIN
              ELSE
                H2(K)=DEY*DEY*ETYMIN*ETYMIN/H2(K)
              END IF
            NPENAL=NPENAL+1
            END IF
  110     CONTINUE
      IF(.NOT.LFIT)GO TO 230
      IF(NPENAL.GT.0)THEN
        DO 220 J=2,N-1
          DO 220 I=1,M
            K=(J-1)*M+I
  220       H1(K)=H1(K-M)+H2(K)
        NITPEN=NITPEN+1
      END IF
      IF(NPENAL.GT.0)GO TO 90
  230 CONTINUE
C
C  Determination of the Y-coordinates of the new gridpoints, using
C  linear interpolation of the ETA-coordinates of the new points
C  The new Y-coordinates are put in the array YN
C
      IF(NHETA.GT.N-2)NHETA=N-2
      NH=(N+1)/2
C     OPEN(22,FILE='WEERBERICHT.',STATUS='NEW')
C     DO 149 J=1,N
C 149 WRITE(22,*)J,H1((J-1)*M+MH)
      DO 160 I=1,M
        YN(I)=0.
        YN((N-2)*M+I)=YMAX/2.
        YN((N-1)*M+I)=YMAX
        YN((NHETA-1)*M+I)=YM(I)
        DO 150 J=2,NHETA-1
          KJ=(J-1)*M+I
          DO 150 JI=2,N-1
            KI=(JI-1)*M+I
            IF(H1(KI-M).LE.ETA(KJ).AND.H1(KI).GT.ETA(KJ))THEN
              YN(KJ)=Y(KI-M)+(ETA(KJ)-H1(KI-M))*(Y(KI)-Y(KI-M))/
     *                (H1(KI)-H1(KI-M))
            END IF
  150     CONTINUE
        DO 155 J=NHETA+1,N-2
          KJ=(J-1)*M+I
          DO 155 JI=2,N-1
```

```
            KI=(JI-1)*M+I
            IF(H1(KI-M).LE.ETA(KJ).AND.H1(KI).GT.ETA(KJ))THEN
              YN(KJ)=Y(KI-M)+(ETA(KJ)-H1(KI-M))*(Y(KI)-Y(KI-M))/
     *                  (H1(KI)-H1(KI-M))
            END IF
  155     CONTINUE
  160   CONTINUE
C     DO 161 J=1,N
C 161 WRITE(22,*)J,YN((J-1)*M+MH)
C     CLOSE(22,STATUS='CRUNCH')
C
C  Changing old and new Y-coordinates.
C
      DO 200 J=1,N
        DO 190 I=1,M
          K=(J-1)*M+I
          YO(K)=Y(K)
          Y(K)=YN(K)
  190 CONTINUE
  200 CONTINUE
C
C  The values of the vorticity in the new gridpoints are interpolated
C
      CALL SHFTOY
C
C  The partial derivatives of KSI and ETA with respect to X and Y are
C  calculated in the subroutine DERIVA
C
      CALL DERIVA
      IF(DIAGNO)THEN
        OPEN(11,TITLE='DIAGNO/XYTOOM.',STATUS='NEW')
        DO 999 J=0,N-1
  999     WRITE(11,*)EX(J*M+MH+1),EY(J*M+MH+1),EXX(J*M+MH+1),
     *                EYY(J*M+MH+1),EXY(J*M+MH+1)
        CLOSE(11,STATUS='CRUNCH')
      END IF
 1000 CONTINUE
      END
C
C***********************************************************************
C
      SUBROUTINE SHFTOX
      INTEGER NI,NK,NJ,K1,K2,K3,K4
      REAL D1,D2,YP,YQ,OP,OQ,XP1,YP1,XQ3,YQ3,X21,Y21,X43,Y43,D2P,D2Q
$INCLUDE'STRIP/BIEB/COMMON'
C
C This subroutine adapts the values of OM in the new gridpoints
C X,Y to the values of OM in the old gridpoints XO,Y
C notice: the line connecting points with equal X-index (i) is a
C straight line. Only the x-coordinates have changed.
C Furthermore the values of OM are zero on the boundaries.
C H1 is a help-array
C First setting value for H1 zero for each gridpoint.
C
      DO 100 K=1,M*N
  100   H1(K)=0.
      DO 200 I=2,M-1
        DO 200 NI=2,M
          IF(XO(NI-1).LE.X(I).AND.XO(NI).GT.X(I))THEN
            DO 150 J=2,N-1
              K=(J-1)*M+I
              DO 150 NJ=2,N
                NK=(NJ-1)*M+NI
                D2=(X(K)-XO(NK-1))/(XO(NK)-XO(NK-1))
                YP=Y(NK-M-1)+D2*(Y(NK-M)-Y(NK-M-1))
                YQ=Y(NK-1)+D2*(Y(NK)-Y(NK-1))
                IF(YP.LE.Y(K).AND.YQ.GT.Y(K))THEN
                  K1=NK-M-1
                  K2=NK-M
                  K3=NK-1
                  K4=NK
                  XP1=X(K)-XO(K1)
                  YP1=YP-Y(K1)
                  XQ3=X(K)-XO(K3)
                  YQ3=YQ-Y(K3)
                  X21=XO(K2)-XO(K1)
                  Y21=Y(K2)-Y(K1)
                  X43=XO(K4)-XO(K3)
                  Y43=Y(K4)-Y(K3)
                  D2P=SQRT((XP1*XP1+YP1*YP1)/(X21*X21+Y21*Y21))
                  D2Q=SQRT((XQ3*XQ3+YQ3*YQ3)/(X43*X43+Y43*Y43))
                  OP=OM(K1)+D2P*(OM(K2)-OM(K1))
                  OQ=OM(K3)+D2Q*(OM(K4)-OM(K3))
                  D1=(Y(K)-YP)/(YQ-YP)
                  H1(K)=OP+D1*(OQ-OP)
                END IF
  150         CONTINUE
            END IF
  200 CONTINUE
      DO 300 J=2,N-1
        DO 300 I=2,M-1
          K=(J-1)*M+I
  300     OM(K)=H1(K)
      END
C
C***********************************************************************
C
      SUBROUTINE SHFTOY
      INTEGER NJ,NK
      REAL D1
```

```
$INCLUDE'STRIP/BIEB/COMMON'
C
C This subroutine adapts the values of OM in the new gridpoints
C X,Y to the values of OM in the old gridpoints X,YO
C notice: the line connecting points with equal X-index (I) is a
C straight line. Only the y-coordinates have changed.
C Furthermore the values of OM are zero on the boundaries.
C Hi is a help-array
C First setting value for Hi zero for each gridpoint.
C
      DO 100 K=1,M*N
  100   HI(K)=0.
      DO 200 I=2,M-1
        DO 200 J=2,N-1
          K=(J-1)*M+I
          DO 200 NJ=2,N
            NK=(NJ-1)*M+I
            IF(YO(NK-M).LE.Y(K).AND.YO(NK).GT.Y(K))THEN
              DI=(Y(K)-YO(NK-M))/(YO(NK)-YO(NK-M))
              HI(K)=OM(NK-M)+DI*(OM(NK)-OM(NK-M))
            END IF
  200 CONTINUE
      DO 300 J=2,N-1
        DO 300 I=2,M-1
          K=(J-1)*M+I
  300     OM(K)=HI(K)
      END
```

STRIP/BIEB/FIE
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:48:03.

```
      REAL FUNCTION FIE(X,PAR,D)
C this function provides the y-value of maximum vorticity for a given x
C
      REAL X,PAR(10),D,X2,PI,PI2,SOM
      INTEGER I,NPAR
      COMMON /PARAM/NPAR
      PI=4.*ATAN(1.)
      X2=X*X
      PI2=PI*PI/4.
      SOM=0.
      DO 10 I=1,NPAR
   10   SOM=SOM+PAR(I)*X/(((2.*I-1)**2)*PI2+X2)
      FIE=D*(1.-2.*SOM)
      END
C***********************************************************************
C
```

STRIP/BIEB/KOPOIS
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:38:42.

```
      SUBROUTINE KOPOIS
C
C  determination of the coefficients for the discretised, transformed
C  Poisson-PSI equation.
C
      REAL  DEXI,DEET
$INCLUDE'STRIP/BIEB/COMMON'
      DEXI=(XMAX-XMIN)/(M-1)
      DEET=YMAX/(N-1)
      DO 100 I=1,L
        AP(I)=ZPT6(I)/(4*DEXI*DEET)
        BP(I)=-ZPT3(I)/(2*DEET)+ZPT5(I)/(DEET**2)
        CP(I)=-AP(I)
        DP(I)=-ZPT2(I)/(2*DEXI)+ZPT4(I)/(DEXI**2)
        EP(I)=-2*(ZPT4(I)/(DEXI**2)+ZPT5(I)/(DEET**2))
        FP(I)=ZPT2(I)/(2*DEXI)+ZPT4(I)/(DEXI**2)
        GP(I)=-AP(I)
        HP(I)=ZPT3(I)/(2*DEET)+ZPT5(I)/(DEET**2)
        IP(I)=AP(I)
        JP(I)=ZPT1(I)
  100 CONTINUE
      END
```

STRIP/BIEB/TRANIP
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:37:20.

```
      SUBROUTINE TRANIP
C
C  transformation of the coefficients of the Poisson-psi equation
C  into the computational plane.
C
$INCLUDE'STRIP/BIEB/COMMON'
      DO 100 I=1,L
        ZPT1(I)=ZPI(I)
        ZPT2(I)=KSIXX(I)
        ZPT3(I)=EXX(I)+EYY(I)
        ZPT4(I)=KSIX(I)**2
        ZPT5(I)=EX(I)**2+EY(I)**2
        ZPT6(I)=2.*KSIX(I)*EX(I)
  100 CONTINUE
      END
```

STRIP/BIEB/RVWPSI
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:39:58.

```
      SUBROUTINE RVWPSI
C
C  this subroutine calculates the effect of the boundary conditions on
C  the coefficients in the differential scheme for the transformed
C  Poisson-PSI equation.
C
      REAL Q(10000),DEXI,DEET,A,Z(100),Z1,Z2
$INCLUDE'STRIP/BIEB/COMMON'
      DEXI=(XMAX-XMIN)/(M-1)
      DEET=YMAX/(N-1)
C
C  boundary conditions for the lower boundary y=0: mirror
C
      DO 100 I=1,M
        Z1=(EY(I)/DEET)**2
        Z2=EYY(I)/(2.*DEET)
        Z(I)=2.*Z1/(Z1-Z2)
  100   Q(I)=(Z1+Z2)/(Z2-Z1)
      DO 200 I=1,M
        IF(I.GT.1)GP(I)=GP(I)+AP(I)*Q(I-1)
        HP(I)=HP(I)+BP(I)*Q(I)
        EP(I)=EP(I)+Z(I)*BP(I)
        IF(I.LT.M)IP(I)=IP(I)+Q(I+1)*CP(I)
        AP(I)=0.
        BP(I)=0.
        CP(I)=0.
  200 CONTINUE
C
C other boundaries:  zero-valued velocities, streamfunction at boundary
C                    and next to boundary equal zero.
C
      DO 300 J=1,N
        DO 300 I=1,M
          K=(J-1)*M+I
          IF(.NOT.(J.GT.0.AND.J.LT.N-1.AND.I.GT.2.AND.I.LT.M-1))THEN
          AP(K)=0.
          BP(K)=0.
          CP(K)=0.
          DP(K)=0.
          EP(K)=1.
          FP(K)=0.
          GP(K)=0.
          HP(K)=0.
          IP(K)=0.
          JP(K)=0.
          END IF
  300 CONTINUE
      END
```

STRIP/BIEB/TRVELO
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:41:30.

```
      SUBROUTINE TRVELO
C
C TRVELO determines the velocities in X- and Y-direction for each grid
C  point. The truncation error of the differential schemes used is of
C  order two.
C
      REAL A,B,DEXI,DEET
$INCLUDE'STRIP/BIEB/COMMON'
      DEXI=(XMAX-XMIN)/(M-1)
      DEET=YMAX/(N-1)
      DO 100 I=1,M
        DO 100 J=1,N
          K=(J-1)*M+I
          IF(I.EQ.1)THEN
            A=(PSI(K+1)-PSI(K))/DEXI
          ELSE IF(I.EQ.M)THEN
            A=(-PSI(K-1)+PSI(K))/DEXI
          ELSE
            A=(PSI(K+1)-PSI(K-1))/(2*DEXI)
          END IF
          IF(J.EQ.1)THEN
            B=(PSI(K+M)-PSI(K))/DEET
          ELSE IF(J.EQ.N)THEN
            B=(-PSI(K-M)+PSI(K))/DEET
          ELSE
            B=(PSI(K+M)-PSI(K-M))/(2*DEET)
          END IF
          U(K)=EY(K)*B
          V(K)=-KSIX(K)*A-EX(K)*B
  100 CONTINUE
      END
```

STRIP/BIEB/REGGRI
DATE & TIME PRINTED: WEDNESDAY, MARCH 9, 1988 @ 20:42:49.

```
      SUBROUTINE REGGRI
C
C  setting up an orthogonal grid with the values of y at x=0 as a base.
C
      INTEGER MH
$INCLUDE'STRIP/BIEB/COMMON'
      MH=(M+1)/2
      DO 10 I=1,M*N
        YO(I)=Y(I)
   10   XO(I)=X(I)
      DO 100 J=1,N
        DO 70 I=1,MH-1
```

```
                K=(J-1)*M+I
     70         Y(K)=Y((J-1)*M+MH)
              DO 100 I=MH+1,M
                K=(J-1)*M+I
     100        Y(K)=Y((J-1)*M+MH)
            END
STRIP/BIEB/COMMON
DATE & TIME PRINTED: MONDAY, FEBRUARY 29, 1988 @ 13:27:53.
```

```
      C   this block contains the variables that are used in all the sub-
      C   routines.
          INTEGER M,N,NT,STOPCR,I,L,J,K,STOPCX,NPAR,NITO,NITP,NCPL,
         *         NITPMX,NITPMY,NOUTQF,NHETA
          REAL XMIN,XMAX,YMAX,D,TBEGIN,TEIND,EPPOME,EPOMMA,EPSIME,EPSIMA,
         * EA,ER,PAR(10),X(10000),Y(10000),U(10000),V(10000),OM(10000),
         * XTIMES,OMO(10000),PSI(10000),KSI(10000),ETA(10000),AO(10000),
         *  BO(10000),CO(10000),DO(10000),EO(10000),FO(10000),GO(10000),
         *  HO(10000),IO(10000),JO(10000),AP(10000),BP(10000),CP(10000),
         *  DP(10000),EP(10000),FP(10000),GP(10000),HP(10000),IP(10000),
         *  JP(10000),
         * ZO2(10000), ZO1(10000),ZO3(10000),ZO4(10000),ZO5(10000),
         * ZOT1(10000),ZOT2(10000),ZOT3(10000),ZOT4(10000),ZOT5(10000),
         * ZOT6(10000),XO(10000),YO(10000),
         * H1(10000),H2(10000),H3(10000),H4(10000),H5(10000),H6(10000),
         * H7(10000),H8(10000),H9(10000),H10(10000),H11(10000),H12(10000),
         * F(10000),ZP1(10000),
         * ZPT1(10000),ZPT2(10000),ZPT3(10000),ZPT4(10000),ZPT5(10000),
         *ZPT6(10000),EX(10000),EXX(10000),EY(10000),EYY(10000),EXY(10000),
         *KSIX(10000),KSIXX(10000),
         *  ALFAO,ALFAP,XM(100),YM(100),WIDTH,RHOO,NU,DT,DAKS,XIXMIN,
         *      XIXMAX,ETYMIN,ETYMAX,PRO
          COMMON /CONFIG/XMIN,XMAX,YMAX,XIXMIN,XIXMAX,ETYMIN,ETYMAX,NCPL,
         *         NITPMX,NITPMY,NHETA
          COMMON /TIME/DT,/CPRO/PRO
          COMMON /DIAM/D,WIDTH,DAKS
          COMMON /XMYM/XM,YM
          COMMON /DERIV/KSIX,KSIXX,EX,EXX,EY,EYY,EXY
          COMMON /COOR/X,Y
          COMMON /SIZE/M,N,L,RHOO,NU
          COMMON /SPEED/U,V
          COMMON /OLDOM/OMO
          COMMON /STRMFU/PSI
          COMMON /COOM/ZO1,ZO2,ZO3,ZO4,ZO5
          COMMON /COPSI/ZP1
          COMMON /FIELD/OM
          COMMON /TRCOOR/KSI,ETA
          COMMON /CODSOM/AO,BO,CO,DO,EO,FO,GO,HO,IO,JO
          COMMON /CODSPS/AP,BP,CP,DP,EP,FP,GP,HP,IP,JP
          COMMON /TRCOOM/ZOT1,ZOT2,ZOT3,ZOT4,ZOT5,ZOT6
          COMMON /TRCOPS/ZPT1,ZPT2,ZPT3,ZPT4,ZPT5,ZPT6
          COMMON /CALPAR/TBEGIN,TEIND,NT,EPOMME,EPOMMA,EPSIME,EPSIMA,
         *              XTIMES,STOPCR,EA,ER,ALFAO,ALFAP
          COMMON /ITER/NITO,NITP
          COMMON /OLDCOR/XO,YO
          COMMON /PARAM/NPAR,PAR
          SUBROUTINE MSTIMP(M,N,OM,AA,BA,CA,DA,EA,FA,GA,HA,IA,JA,A,B,C,
         *          D,E,F,G,H,U,DE,VH,R,ALFA,EPSMEA,EPSMAX,NITMAX,NITTOT)
      C
      C THIS SUBROUTINE GIVES A SOLUTION TO A NINE POINT MOLECULE DIFFERENCE
      C SCHEME BY APPLICATION OF THE MODIFIED STRONGLY IMPLICIT PROCEDURE,
      C AS DESCRIBED IN : SCHNEIDER AND ZEDAN, NUMERICAL HEAT TRANSFER,
      C VOL. 4, PP.1-19, 1981.
      C
          INTEGER M,N,I,J,K,L,NIT,NITMAX
          REAL OM(M*N),AA(M*N),BA(M*N),CA(M*N),DA(M*N),EA(M*N),FA(M*N),
         *     GA(M*N),HA(M*N),IA(M*N),JA(M*N),A(M*N),B(M*N),C(M*N),D(M*N),
         *     E(M*N),F(M*N),G(M*N),H(M*N),U(M*N),DE(M*N),R(M*N),ALFA,
         *     VH(M*N),FI1,FI2,FI3,FI4,EPSMEA,EPSMAX
          REAL NORM,MAXN,DEK2
          L=M*N
          NIT=0
      C
      C  CALCULATION OF THE COEFFICIENTS OF THE L- AND U-MATRICES
      C
          DO 50 J=1,N
            DO 50 I=1,M
              K=(J-1)*M+I
              IF(I.EQ.1.OR.J.EQ.1)THEN
                A(K)=0.
                ELSE
                A(K)=AA(K)
              END IF
              IF(J.EQ.1)THEN
                B(K)=0.
              ELSE
                B(K)=BA(K)
                IF(I.GT.1)B(K)=B(K)-A(K)*F(K-M-1)
                IF(I.NE.M)B(K)=B(K)-ALFA*CA(K)*F(K-M+1)
              B(K)=B(K)/(1.-ALFA*F(K-M)*F(K-M+1))
              END IF
              IF(I.EQ.M.OR.J.EQ.1)THEN
                C(K)=0.
              ELSE
                C(K)=CA(K)-B(K)*F(K-M)
              END IF
              IF(I.EQ.1)THEN
                D(K)=0.
              ELSE
                D(K)=DA(K)
                IF(J.GT.1)D(K)=D(K)-A(K)*H(K-M-1)-B(K)*G(K-M)-
         *                    2.*ALFA*A(K)*G(K-M-1)
```

```
        D(K)=D(K)/(1.+2.*ALFA*G(K-1))
      END IF
      IF(I.EQ.M.OR.J.EQ.1)THEN
        FI1=0.
      ELSE
        FI1=C(K)*F(K-M+1)
      END IF
      IF(I.EQ.1.OR.J.EQ.1)THEN
        FI2=0.
      ELSE
        FI2=A(K)*G(K-M-1)
      END IF
      IF(I.EQ.M.OR.J.EQ.1)THEN
        FI3=0.
      ELSE
        FI3=C(K)*U(K-M+1)
      END IF
      IF(I.EQ.1)THEN
        FI4=0.
      ELSE
        FI4=D(K)*G(K-1)
      END IF
      E(K)=EA(K)+ALFA*(FI1+FI2+FI3+2.*FI4)
      IF (I.GT.1.AND.J.GT.1)E(K)=E(K)-A(K)*U(K-M-1)
      IF(J.GT.1)E(K)=E(K)-B(K)*H(K-M)
      IF(I.LT.M.AND.J.GT.1)E(K)=E(K)-C(K)*G(K-M+1)
      IF(I.GT.1)E(K)=E(K)-D(K)*F(K-1)
      IF(I.EQ.M)THEN
        F(K)=0.
      ELSE
        F(K)=FA(K)-2.*ALFA*(FI1+FI3)
        IF(J.GT.1)F(K)=F(K)-B(K)*U(K-M)-C(K)*H(K-M+1)
      END IF
      F(K)=F(K)/E(K)
      IF(I.EQ.1.OR.J.EQ.N)THEN
        G(K)=0.
      ELSE
        G(K)=GA(K)
        IF(I.GT.1)G(K)=G(K)-D(K)*H(K-1)
        G(K)=G(K)/E(K)
      END IF
      IF(J.EQ.N)THEN
        H(K)=0.
      ELSE
        H(K)=HA(K)-ALFA*FI4
        IF(I.GT.1)H(K)=H(K)-D(K)*U(K-1)
        H(K)=H(K)/E(K)
      END IF
      IF(J.EQ.N.OR.I.EQ.M)THEN
        U(K)=0.
      ELSE
        U(K)=IA(K)/E(K)
      END IF
   50 CONTINUE
C
  100 CONTINUE
C BEGINNING OF THE LOOP
C
      NIT=NIT+1
C
C  CALCULATION OF THE RESIDUAL VECTOR Rj=JAj-TkjOMk
C
      DO 200 K=1,M*N
          R(K)=JA(K)-EA(K)*OM(K)
          IF(K.GT.1)     R(K)=R(K)-DA(K)*OM(K-1)
          IF(K.GT.M+1)  R(K)=R(K)-AA(K)*OM(K-M-1)
          IF(K.LT.L-M+1)R(K)=R(K)-GA(K)*OM(K+M-1)
          IF(K.GT.M)     R(K)=R(K)-BA(K)*OM(K-M)
          IF(K.GT.M-1)  R(K)=R(K)-CA(K)*OM(K-M+1)
          IF(K.LT.L-1)  R(K)=R(K)-FA(K)*OM(K+1)
          IF(K.LT.L-M-1)R(K)=R(K)-IA(K)*OM(K+M+1)
          IF(K.LT.L-M)  R(K)=R(K)-HA(K)*OM(K+M)
  200 CONTINUE
C
C  CALCULATION OF VH, THE RESULT OF LVH=R
C
      VH(1)=R(1)/E(1)
      DO 300 K=2,L
        VH(K)=R(K)-D(K)*VH(K-1)
        IF(K.GT.M-1)VH(K)=VH(K)-C(K)*VH(K-M+1)
        IF(K.GT.M)  VH(K)=VH(K)-B(K)*VH(K-M)
        IF(K.GT.M+1)VH(K)=VH(K)-A(K)*VH(K-M-1)
        VH(K)=VH(K)/E(K)
  300 CONTINUE
C
C  CALCULATION OF THE DIFFERENCE VECTOR DE BY U.DE=VH, AND OF THE
C  NEW SOLUTION VECTOR OM=OM+DE
C
      NORM=0.
      MAXN=0.
      DE(L)=VH(L)
      OM(L)=OM(L)+DE(L)
      NORM=ABS(DE(L))
      MAXN=NORM
      DO 400 K=L-1,1,-1
        DE(K)=VH(K)-F(K)*DE(K+1)
        IF(K.LT.L-M+1)DE(K)=DE(K)-G(K)*DE(K+M-1)
        IF(K.LT.L-M)  DE(K)=DE(K)-H(K)*DE(K+M)
        IF(K.LT.L-M-1)DE(K)=DE(K)-U(K)*DE(K+M+1)
        OM(K)=OM(K)+DE(K)
        DEK2=ABS(DE(K))
        IF(DEK2.GT.MAXN)MAXN=DEK2
        NORM=NORM+DEK2
  400 CONTINUE
      NORM=NORM/L
      IF(NORM.LE.EPSMEA.AND.MAXN.LE.EPSMAX)GO TO 500
      IF(NIT.GE.NITMAX)GO TO 500
      GO TO 100
  500 CONTINUE
      NITTOT=NITTOT+NIT
      END
```