"KAPPA" – KARLSRUHE PARALLEL PROGRAM FOR AERODYNAMICS

FRANCO MAGAGNATO

Institute for Fluid Dynamics
University of Karlsruhe (TH)
Kaiserstrasse 12, D-76131 Karlsruhe

1. Introduction

The research in fluid dynamics can be done on both experimental and numerical basis. For the latter one a computer code needs to be written in order to solve the governing fluid flow equations. The code KAPPA is a CFD-simulation package serving as a platform to develop faster and more accurate numerical schemes, better physical models, or as an engineering tool for the simulations of flows in technical equipment. Another important subject is the training and education of students or engineers. Since CFD is highly calculation intensive, new computer architectures like vector and parallel computers are necessary to treat more complex flow fields or to resolve these flows more accurately . Therefore KAPPA has been specially designed to be used on these architectures. The structure of the code is such that the solution of additional transport equations needed for the simulation of chemistry , turbulence modeling, multiphase flows etc. can be easily implemented. In order to treat complex geometries the code is block stuctured. The finite volume method is used to discretize the equations in space. The code is written in Fortran 90 using the highly desirable new feature of this language in order to make up for the disadvantages of the Fortran 77 language with respect to other programming languages like C, C++, or Pascal. For the application of the code on parallel computers a message passing tool has been used. Since we feel that MPI (Message Passing Interface) has become the defacto standard tool for distributed memory parallel platform, it is used in KAPPA. Especially for the training of students in CFD, a graphical user interface for KAPPA has been developed. This has been done using the freeware Tcl/Tk (Tool command language/Tool kit) developed by John Ousterhout [19] which has gained widespread acceptance in the programming community. It turned out that with the graphical user interface not only the handling of the code has been improved but also the amount of typing errors has been significantly

reduced. The code solves the compressible Navier-Stokes equations in Reynolds average form in conjunction with a statistical turbulence model like algebraic turbulence model or two-equation model, as well as in a space and time filtered form for large eddy simulation.

Steady and unsteady flow fields can be simulated. Unsteady flows can also be treated together on moving grids.

The code is subject to continuous development with respect to compressible flow fields, chemically reacting flows, multi phase flows, turbulence modelling etc. in our institute as well as in other research facilities in Europe.

2. Basic Equations Governing Fluid Flow

The calculation of viscous flow around complex 3D-bodies with large seperation regions requires the solution of the Navier-Stokes equations. An exact solution for this system of nonlinear and coupled partial differential equations in conjunction with geometric and dynamic boundary conditions to which the system may be subjected is usually difficult to obtain. One possibility to solve this system is the direct simulation of the full Navier-Stokes equations with a numerical scheme which resolve all the significant time and space scales appearing in a turbulent flow. The effort to solve a flow field increases approximately proportional, with the third power of the Reynolds-number. Even with the help of the highest sophisticated super computers available today and in the near future the direct simulation method can only be used for Reynolds-number regions in the order $0(10^4)$ which is definitively too low for practical flow situations.

In order to obtain the governing conservation equation for turbulent flows for high Reynolds-numbers it is convenient to split the instantaneous quantities of the Navier-Stokes equations into a mean and fluctuating part. This concept is called the Reynolds averaging technique.

The average of a turbulence variable can be defined in several ways, e.g. time, mass, phase or ensemble averaging [33]. The appropriate averaging concept for compressible and stationary flows is the mass-weighted averaging after Favre [3]. Replacing the instantaneous quantities in the Navier-Stokes equations by their mean and fluctuating parts results in an expression which contains mean terms, fluctuating terms and additional unknown terms representing the mean effects of turbulence [18]. The additional terms make the resulting conservation equations undetermined, the governing equations do not form a closed set. They require additional relations, based on statistical or similarity considerations. The unknown Terms — $\overline{\rho v_i' v_j'}$ are called Reynolds stresses and are subject to turbulence modeling techniqes.

The compressible, time-dependent Navier-Stokes-equations in integral form can be written as:

for mass conservation:

$$\iiint_V \frac{\partial \rho}{\partial t} dV + \iint_S \rho \vec{v} \cdot \vec{n} \, dS = 0 , \tag{2.1}$$

for momentum conservation:

$$\iiint_V \frac{\partial(\rho\vec{v})}{\partial t} dV + \iint_S \rho\vec{v}(\vec{v}\cdot\vec{n})dS + \iint_S p\vec{n}\, dS - \iint_S \vec{n}\cdot\overline{T}\, dS = 0, \tag{2.2}$$

for energy conservation:

$$\iiint_V \frac{\partial(\rho E)}{\partial t} dV + \iint_S \rho E(\vec{v}\cdot\vec{n})dS + \iint_S p(\vec{v}\cdot\vec{n})dS - \iint_S v(\vec{n}\cdot\overline{T})dS + \iint_S \vec{n}\cdot\vec{q}\, dS = 0. \tag{2.3}$$

This is a system of hyperbolic equations with respect to time, where E represents the total specific energy (summation of inner and kinetic energy) and $\vec{q}$ being the energy flux vector.

It is assumed that the energy flux vector expresses only molecular energy transport which can be described by Fourier's law:

$$\vec{q} = -k\nabla\overline{T}. \tag{2.4}$$

With k the thermal conductivity, determined with the assumption of a constant Prandtl number according to:

$$Pr = \frac{c_p\mu}{k}. \tag{2.5}$$

$\overline{T}$ in the momentum equation is the „viscous stress tensor" which will be described later.

The total specific energy for ideal gas is:

$$\rho E = \rho e + \rho/2\vec{v}^2 = \rho/(\kappa - 1) + \rho/2\vec{v}^2 \tag{2.6}$$

and the total specific enthalpy:

$$\rho H = \rho E + p. \tag{2.7}$$

Here e represents the inner energy, H the total enthalpy and κ the ratio of the specific heats.

For Newton-type fluids the assumption is made, that the stress tensor $\overline{T}$ is continually varying with deformation velocity tensor $\overline{D}$ (Stokes hypothesis).

$$\overline{T} = 2\mu\overline{D} - \frac{2}{3}\mu\nabla\cdot(\vec{v}\cdot\vec{I}) \tag{2.8}$$

The deformation velocity tensor $\overline{D}$ is given by

$$\overline{D} = \begin{bmatrix} u_x & 1/2(u_y + v_x) & 1/2(u_z + w_x) \\ 1/2(u_y + v_x) & v_y & 1/2(v_z + w_y) \\ 1/2(u_z + w_x) & 1/2(v_z + w_y) & w_z \end{bmatrix} \tag{2.9}$$

μ being the total viscosity (laminar + turbulent) and I being the unit-tensor. Furthermore, the dependency of the laminar viscosity μ_l from the pressure is neglected so that the Sutherlan formulation could be used.

$$\mu_l = \frac{c\sqrt{T}}{1 + \frac{d}{T}} \tag{2.10}$$

with c and d being fluid specific constants which for air is given by:

$$c = 1.46 \cdot 10^{-6} \quad \left[\frac{kg}{ms\sqrt{K}}\right]$$
$$d = 110.4K$$

3. Finite Volume Method

There are some different approaches for the approximation of the equations governing fluid flows. The most popular are the Finite Difference, Finite Element and Finite Volumes Methods. All methods have their particular advantages and disadvantages, see Ferziger/Perić [37] or Hirsch [9] for a detailed description.

In KAPPA we are using the Finite Volume Method to approximate the integral form of the conservation equations. The solution domain is subdivided into a finite number of contigous control volumes (CV), and the conservation equations, as well as the transport equations for turbulence, species etc. are applied to each CV. At the centroid of each CV lies a computational node at which the variable values are to be calculated. Interpolation is used to express variable values at the CV surface in terms of the nodal (CV-center) values. Surface and volume integrals are approximated using suitable quadrature formulae.

As a result, one obtains an algebraic equation for each CV , in which a number of neighbour nodal values appears. In this chapter we shall deal mostly with 2D grids; the extension to 3D-problems is straight forward. We will describe the method for an arbitrary conservation equation ϕ on a grid moving with the velocity $\vec{v}_g$

$$\frac{\partial}{\partial t}\int\int\int_V \rho\phi\, dV + \int\int_S \rho\phi\left(\vec{v} - \vec{v}_g\right)\cdot\vec{n}\, dS =$$
$$\int\int_S \Gamma\, grad\ \phi \cdot \vec{n}\, dS + \int\int\int_V q_\phi\, dV \tag{3.1}$$

Dividing the flow field in a finite number of control volumes one needs to approximate the volume and the surface integral.

The simplest approximation of the integral is the midpoint rule: the integral is approximated as a product of the integrand and the volume or surface area (this results in second order accuracy). For the volume integrals one assumes a const. value of the integrand and multiply it with the volume. The surface integral is approximated in terms of the variable values at one or more locations on the cell face and the cell face values are interpolated in terms of the nodal values.

There are several ways of doing this, the simplest one is to take the value at the node upstream of the face.

For example on the face e one has to check the flow direction at this face and then to take either:

$$\phi_e = \begin{cases} \phi_P & \text{if } \left((\vec{v} - \vec{v}_g)\cdot \vec{n}\right)_e > 0 \\ \phi_E & \text{if } \left((\vec{v} - \vec{v}_g)\cdot \vec{n}\right)_e < 0 \end{cases}. \tag{3.2}$$

This approximation is only first order accurate — which is regarded as insufficient for practical calculations, therefore a second order approximation is necessary for the face value ϕ_e

$$\phi_e = \frac{\phi_E + \phi_P}{2}. \tag{3.3}$$

For equidistant grids this approximation is second order accurate whereby the formulation:

$$\phi_e = \lambda_e \phi_E + (1 - \lambda_e)\phi_P, \tag{3.4}$$

where linear interpolation factor λ is defined as:

$$\lambda_e = \frac{\vec{x}_e - \vec{x}_P}{\vec{x}_E - \vec{x}_P} \tag{3.5}$$

is second order accurate on non-equidistant grids [37].

4. Higher Order Approximations

The most critical terms in the Navier-Stokes equation are the convective fluxes since they are nonlinear. If one wants to apply higher order approximations usually only these terms are treated differently .

In Finite Volume Methods one has to approximate then the surface integral with three points in 2D and with 9 points in 3D for fourth-order accuracy according to Simpson's rule. In order to retain the fourth-order accuracy these values have to be obtained by interpolation of the nodal values at least as accurate as Simpson's rule.

Cubic polynomials are suitable by fitting

$$\phi_{(x)} = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{4.1}$$

through the values of ϕ at four nodes (two on either side of the face).

5. Turbulence Models

Turbulence modeling concerns the generation and testing of closure relations describing the Reynold stresses. At present we can confidently predict only a small class of turbulent flows. Turbulence models can be devided into three major kategories: eddy-viscosity models, algebraic Reynolds stress models and differential Reynolds stress models. The differential Reynolds stress models consist of partial differential equation for each component of the Reynolds stresses $\overline{\rho v_i' \cdot v_j'}$. They can be derived in exact forms but contain higher order correlation that have to be approximated in order to obtain a closed system. In these models one needs to solve the equation for the turbulence energy dissipation rate ε, in addition to those for $\overline{\rho v_i' \cdot v_j'}$ for the length scale. A particular advantage of the Reynolds stress models is that terms accounting for buoyancy, rotation and other effects are in principle introduced automatically. Several closure schemes have been proposed by Launder et al. [40], Lumley [17], Gibson/Rodi [39] and others.

In differential stress models, there are differential transport equations for each component of $\overline{\rho v_i' \cdot v_j'}$ in addition to the ε-equation. To reduce the computational effort, Rodi [50] proposed an algebraic relation to calculate the Reynold stresses. The convection and diffusion terms in the transport equation of $\overline{\rho v_i' \cdot v_j'}$ are replaced by model approximations, reducing the equations to algebraic equations. Rodi assumes that the transport of $\overline{\rho v_i' \cdot v_j'}$ is proportional to the transport of K and the proportionality factor is $\frac{\overline{\rho v_i' \cdot v_j'}}{K}$. With these approximations incorporated, the transport equations yield algebraic expressions for $\overline{\rho v_i' \cdot v_j'}$ that contain the various production terms appearing in the $\overline{\rho v_i' \cdot v_j'}$ equations. Thus, the gradient of mean flow quantites, K and ε appears also in the expression, so that K and ε equations have to be added in order to complete the turbulence model. The algebraic expression together with the K and ε equation form an extended K-ε model. Algebraic stress models are suitable whenever the transport of $\overline{\rho v_i' \cdot v_j'}$ is not important. Algebraic stress relations are basically like

eddy viscosity formulations [51] and therefore are not applicable to cases with counter gradient transport. On the other hand, all effects that enter the transport equations for $\overline{\rho v_i' \cdot v_j'}$ through the source terms for example, body force effects (bouyancy , rotation and streamline curvature), non-isotropic strain fields and wall damping influence can be incorporated. So algebraic stress models can also simulate many of the flow phenomena that were described successfully by differential Reynolds stress models.

Boussinesq [12] was the first to attack the problem of finding a model for the Reynolds stresses by introducing the concept of eddy viscosity . He assumed that the turbulent stresses act like the viscosity stress, which implies that the turbulent stresses are proportional to the velocity gradient. The coefficient of proportionality was called the "eddy viscosity" and was defined by

$$-\overline{\rho v_i' \cdot v_j'} = \mu_t \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_i}{\partial x_j} - \frac{2}{3}\delta_{ij} \frac{\partial v_i}{\partial x_j} \right) - \frac{2}{3}\delta_{ij}\rho k \,. \tag{5.1}$$

Note that this viscosity is a property of the fluid motion and not a physical property of the fluid itself. From dimensional considerations and by analogy with kinetic theory the eddy viscosity is proportional to the product of a length and a velocity scale. Many types of eddy viscosity models have been proposed in the past. We will restrict ourselves to the description of those models which have been introduced in the code.

The simplest turbulence models are the algebraic eddy viscosity models which relate the eddy viscosity μ_t by an algebraic expression for the length and velocity scale.

5.1 Baldwin/Lomax model

The Baldwin/Lomax model [34] calculates the eddy viscosity μ_t algebraically from mean flow quantities introducing a two layer concept based on the idea of Cebeci/ Smith [42] and deviding the flow into an inner and outer layer . For the inner layer they assumed:

$$\mu_t = \rho l^2 |\omega| \,, \tag{5.2}$$

where

$$l = k y \left[1 - e^{-y^+ / A^+} \right], \tag{5.3}$$

y = normal distance from the wall

$$y^+ = \frac{\sqrt{\rho_w \tau_w} y}{\mu_w} \,, \tag{5.4}$$

and

$$|\omega| = \sqrt{\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right)^2}. \tag{5.5}$$

The constants $k = 0.4$ and $A^+ = 26$ according to van Driest.

For the outer layer:

$$\mu_t = kC_{cp}\rho F_{wake} F_{kleb}(y) \tag{5.6}$$

with

$$F_{wake} = min\left(y_{max}F_{max}, \frac{C_{wk}y_{max}U_{diff}^2}{F_{max}}\right) \tag{5.7}$$

where y_{max} and F_{max} are determined by the maximum of the folowing function:

$$F(y) = y|\omega|\left[1 - e^{-y^+/A^+}\right] \tag{5.8}$$

This function has a pronounced maximum in the boundary layer. The normal distance from the wall to this point (y_{max}) replaces the displacement thickness in the formulation of Cebeci/Smith. The Klebanoff-factor is given by:

$$F_{kleb}(y) = \frac{1}{1 + 5.5\left(\frac{C_{kleb}y}{y_{max}}\right)^6} \tag{5.9}$$

and U_{diff} is the maximum of the velocity in the wall layer. The other constants are given below:

K = 0.0168 (Klauser);

$c_{cp} = 1.6$; $C_{wk} = 0.25$; $C_{kleb} = 0.3$.

Switching from the inner to the outer model is performed where values of μ_t for inner and outer model become equal.

5.2 Martinelli/Yakhot model

The algebraic turbulence model of Martinelli/Yakhot [48] is based on the Renormalisation Group Theory [36]. Although free from uncertainties related to the determination of modelling constants, they still require the specification of a length scale which leads to a restriction in the generality of the model. The eddy-viscosity is obtained from the following relation.

$$\mu = \mu_l \left[1. + H\left(\frac{a}{\mu_l^3} \varepsilon \Lambda_f^{-4} \right) - C_c \right] 1/3, \tag{5.10}$$

where $H(x)$ is the Heavyside function defined by $H(x) = x$ for $x \geq 0$ and $H(x) = 0$ otherwise, Λ_f the wave vector corresponding to the integral scale of the turbulence in the inertial range, and $\mu = \mu_l + \mu_t$ is the total viscosity (laminar plus turbulent). The constants $a \approx 0.12$ and $C_c \approx 75$ were derived in [36]. The mean dissipation rate ε and the wave vector Λ_f must be determined before using this equation to compute the eddy viscosity.

The integral scale of turbulence $L_f = \pi \Delta_f^{-1}$, corresponding to the top of the inertial range, is postulate to be proportional to the distance from the wall y ($L_f = \kappa y$, where κ is the Von Karman constant).

In the outer region, it is plausible to take the integral scale in the order of the boundary layer thickness δ. Following the idea of Stock/Haase [28] the boundary layer thickness δ is determined by:

$$\delta = 1.548 y_{max}$$

where y_{max} is the wall distance where the function

$$F_{(y)} = y|\omega| \left[1 - e^{-y^+/A^+} \right] \tag{5.11}$$

has its maximum.

With this assumption the eddy viscosity equation can be cast in the following form:

$$\mu = \mu_l \left[1. + H\left(\frac{a}{\mu_l^3} \varepsilon \left(\frac{1}{y} + \frac{1}{\gamma \delta} \right)^{-4} \right) - C_c \right]^{1/3} \tag{5.12}$$

where $a = 0.192$ or $a = 0.0256$, depending whether the Von Karman is taken to be equal to the value predicted by the RNG theory ($\kappa = 0.372$) or to the standard value ($\kappa = 0.4$). The value of the parameter $\gamma = 0.225$ has been chosen in order to recover the constant predicted by the RNG-theory for the outer part of the boundary layer. More precisely the value of γ has been chosen in such a way that $\gamma a^{1/4} \delta \to 0.084 \delta$ as $\gamma \to \delta$. Also equilibrium (Production = Dissipation) is assumed, from which follows:

$$\varepsilon = P_k = \mu_t \xi = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \frac{\partial u_k}{\partial x_k} \right) \frac{\partial u_i}{\partial xj}. \tag{5.13}$$

The turbulent eddy-viscosity μ_t is then obtained by solving the cubic equation at every point in the computational domain.

5.3 Algebraic wake model

In order to use the simple algebraic models on block-structured grids, the problem arises how to calculate the turbulent stresses in those blocks which do not have a boundary-layer-type flow. A typical example is the wake of blunt bodies, because the algebraic models cannot calculate the characteristic length and the velocity scale normal to the wall. Therefore, empirical distribution laws were introduced to describe the turbulence transport into these blocks. Because the production of turbulent energy in local equilibrium flows correlates with vorticity, this was taken as a weighted function to distribute eddy-viscosity in that area, according to

$$\mu_t' = \mu_{t_v} \left(\frac{|\omega|}{|\omega_v|} \right)^{\alpha}, \tag{5.14}$$

with: μ_{t_v} – the maximum eddy-viscosity along the upstream block face and $|\omega_v|$ – the local vorticity at that point. The exponent α is determined by numerical experiments, $\alpha = 0.2$.

In addition the eddy-viscosity is also smoothed by an exponential damping factor in order to ensure steadiness on the block faces.

$$\mu_t = \mu_t' - \left(\mu_t' - \mu_{t_j} \right) \exp\left(-\frac{\Delta x_j}{\Delta x_{\max}} \right), \tag{5.15}$$

with: μ_{t_j} – the eddy-viscosity on the upstream blockface at the same j-station as μ_t', Δx_{max} – the normal distance to the location of μ_t' and Δx_{max} – the length of the block in streamwise direction.

5.4 K-ε model of Launder and Sharma

The most popular two-equation model is the K-ε model, with K the turbulent kinetic energy and ε the dissipation rate of the turbulent kinetic energy. These two equations have been derived from the full Navier-Stokes equations by adopting the decomposition of the instantaneous variables in a mean part and a fluctuating part according to Reynolds. Additional correlations appear from the non-linear part of the Navier-Stokes equations, which must be modelled appropriately [43].

The K-ε turbulence model is valid only in the high-Reynolds number region. In order to use the two-equation models throughout the laminar, transition and fully turbulent regions one must extend these models to account for the wall proximity effects. Patel et al. [44] reviewed eight different models and found that the model of Launder and Sharma [41] appears to perform fairly well in a majority of the test cases studied by Patel et al. [44].

The full K-ε model in generalized curvilinear coordinates reads:

$$\frac{\partial W}{\partial t}+\frac{\partial F}{\partial \xi}+\frac{\partial G}{\partial \eta}+\frac{\partial H}{\partial \zeta}=S \tag{5.16}$$

with

$$W=\begin{bmatrix}\rho K Vol\\ \rho \varepsilon Vol\end{bmatrix} \tag{5.17}$$

$$F=\begin{bmatrix}\rho K\overline{U}-\dfrac{\mu_K}{Vol}\left(c_1\dfrac{\partial K}{\partial \xi}+c_2\dfrac{\partial K}{\partial \eta}+c_3\dfrac{\partial K}{\partial \zeta}\right)\\ \rho \varepsilon \overline{U}-\dfrac{\mu_\varepsilon}{Vol}\left(c_1\dfrac{\partial \varepsilon}{\partial \xi}+c_2\dfrac{\partial \varepsilon}{\partial \eta}+c_3\dfrac{\partial \varepsilon}{\partial \zeta}\right)\end{bmatrix} \tag{5.18}$$

$$G=\begin{bmatrix}\rho K\overline{V}-\dfrac{\mu_K}{Vol}\left(c_4\dfrac{\partial K}{\partial \xi}+c_5\dfrac{\partial K}{\partial \eta}+c_6\dfrac{\partial K}{\partial \zeta}\right)\\ \rho \varepsilon \overline{V}-\dfrac{\mu_\varepsilon}{Vol}\left(c_4\dfrac{\partial \varepsilon}{\partial \xi}+c_5\dfrac{\partial \varepsilon}{\partial \eta}+c_6\dfrac{\partial \varepsilon}{\partial \zeta}\right)\end{bmatrix} \tag{5.19}$$

$$H=\begin{bmatrix}\rho K\overline{W}-\dfrac{\mu_K}{Vol}\left(c_7\dfrac{\partial K}{\partial \xi}+c_8\dfrac{\partial K}{\partial \eta}+c_9\dfrac{\partial K}{\partial \zeta}\right)\\ \rho \varepsilon \overline{W}-\dfrac{\mu_\varepsilon}{Vol}\left(c_7\dfrac{\partial \varepsilon}{\partial \xi}+c_8\dfrac{\partial \varepsilon}{\partial \eta}+c_9\dfrac{\partial \varepsilon}{\partial \zeta}\right)\end{bmatrix} \tag{5.20}$$

$$S = \begin{bmatrix} Vol\left(P_S + P_K - \rho\varepsilon - 2\rho\nu\left(\frac{\partial\sqrt{K}}{\partial x_j} \right)^2 \right) \\ Vol\left(c_1' \frac{\varepsilon}{K}\left[P_S + P_K\right] - \rho\, c_2' \frac{\varepsilon^2}{K} \cdot f_2 - 2\mu_l\mu_t\left(\frac{\partial^2 U_i}{\partial x_i x_j} \right)^2 \right) \end{bmatrix} \tag{5.21}$$

$$\overline{U} = u \cdot S_i^x + \upsilon \cdot S_i^y + w \cdot S_i^z \tag{5.22}$$

$$\overline{V} = u \cdot S_j^x + \upsilon \cdot S_j^y + w \cdot S_j^z \tag{5.23}$$

$$\overline{W} = u \cdot S_k^x + \upsilon \cdot S_k^y + w \cdot S_k^z \tag{5.24}$$

$$c_1 = \left(S_i^x\right)^2 + \left(S_i^y\right)^2 + \left(S_i^z\right)^2 \qquad c_5 = \left(S_j^x\right)^2 + \left(S_j^y\right)^2 + \left(S_j^z\right)^2 \tag{5.25}$$

$$c_2 = c_4 = S_i^x S_j^x + S_i^y S_j^y + S_i^z S_j^z \qquad c_8 = c_6 = S_j^x S_k^x + S_j^y S_k^y + S_j^z S_k^z \tag{5.26}$$

$$c_3 = c_7 = S_i^x S_k^x + S_i^y S_k^y + S_i^z S_k^z \qquad c_9 = \left(S_k^x\right)^2 + \left(S_k^y\right)^2 + \left(S_k^z\right)^2 \tag{5.27}$$

$$P_K = -\frac{2}{3}\rho\, K\left[\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z} \right] \tag{5.28}$$

$$P_S = \left[2\left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 + \left(\frac{\partial w}{\partial z} \right)^2 \right] + \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right]^2 + \left[\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right]^2 + \left[\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right]^2 - \frac{2}{3}\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right]^2 \right] \tag{5.29}$$

with

$$\frac{\partial u}{\partial x} = \left(\frac{\partial u}{\partial \xi} S_i^x + \frac{\partial u}{\partial \eta} S_j^x + \frac{\partial u}{\partial \zeta} S_k^x \right) \frac{1}{vol} \tag{5.30}$$

$$\frac{\partial u}{\partial y} = \left(\frac{\partial u}{\partial \xi} S_i^y + \frac{\partial u}{\partial \eta} S_j^y + \frac{\partial u}{\partial \zeta} S_k^y \right) \frac{1}{vol} \tag{5.31}$$

$$\frac{\partial u}{\partial z} = \left(\frac{\partial u}{\partial \xi} S_i^z + \frac{\partial u}{\partial \eta} S_j^z + \frac{\partial u}{\partial \zeta} S_k^z \right) \frac{1}{vol} \tag{5.32}$$

etc.

$$\mu_K = (\mu_l + \lambda_K \mu_t)$$

$$\begin{array}{lll} c_1' = 1.44 & c_2' = 1.92 & c_\mu = 0.09 \\ c_\varepsilon = 0.491 & \lambda_\varepsilon = 0.77 & \lambda_K = 1.0 \end{array} \tag{5.33}$$

$$\mu_\varepsilon = (\mu_l + \lambda_\varepsilon \mu_t) \tag{5.34}$$

$$R_T = \frac{\rho K^2}{\mu_l \varepsilon} \tag{5.35}$$

$$\mu_t = \mu_l c_\mu f_\mu R_T \tag{5.36}$$

$$f_2 = 1 - 0.3 \exp\left(-R_T^2\right) \tag{5.37}$$

$$f_\mu = \exp\left(\frac{-2.5}{1 + 0.02 R_T} \right) \tag{5.38}$$

5.5 K-τ model of Speziale et al.

Very recently Speziale et al. [27] proposed a K-τ model with improved asymptotic behaviour of the wall damping functions. This model yields improved predictions for turbulent boundary layers and is computationally robust. This robustness makes this model very attractive to use in a 3D-Navier-Stokes solver for complex geometries.

The K-τ model proposed by Speziale et al. in generalized curvilinear coordinates reads:

$$\frac{\partial W}{\partial t} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} + \frac{\partial H}{\partial \zeta} = S \tag{5.39}$$

with

$$W = \begin{bmatrix} \rho\, KVol \\ \rho\tau\ \mathrm{Vol} \end{bmatrix} \tag{5.40}$$

$$F = \begin{bmatrix} \rho\, K\overline{U} - \dfrac{\mu_K}{Vol}\left(c_1 \dfrac{\partial K}{\partial \xi} + c_2 \dfrac{\partial K}{\partial \eta} + c_3 \dfrac{\partial K}{\partial \zeta} \right) \\ \rho\tau\, \overline{U} - \dfrac{\mu_\tau}{Vol}\left(c_1 \dfrac{\partial \tau}{\partial \xi} + c_2 \dfrac{\partial \tau}{\partial \eta} + c_3 \dfrac{\partial \tau}{\partial \zeta} \right) \end{bmatrix} \tag{5.41}$$

$$G = \begin{bmatrix} \rho\, K\overline{V} - \dfrac{\mu_K}{Vol}\left(c_4 \dfrac{\partial K}{\partial \xi} + c_5 \dfrac{\partial K}{\partial \eta} + c_6 \dfrac{\partial K}{\partial \zeta} \right) \\ \rho\tau\, \overline{V} - \dfrac{\mu_\tau}{Vol}\left(c_4 \dfrac{\partial \tau}{\partial \xi} + c_5 \dfrac{\partial \tau}{\partial \eta} + c_6 \dfrac{\partial \tau}{\partial \zeta} \right) \end{bmatrix} \tag{5.42}$$

$$H = \begin{bmatrix} \rho\, K\overline{W} - \dfrac{\mu_K}{Vol}\left(c_7 \dfrac{\partial K}{\partial \xi} + c_8 \dfrac{\partial K}{\partial \eta} + c_9 \dfrac{\partial K}{\partial \zeta} \right) \\ \rho\tau\, \overline{W} - \dfrac{\mu_\tau}{Vol}\left(c_7 \dfrac{\partial \tau}{\partial \xi} + c_8 \dfrac{\partial \tau}{\partial \eta} + c_9 \dfrac{\partial \tau}{\partial \zeta} \right) \end{bmatrix} \tag{5.43}$$

$$S = \begin{bmatrix} Vol\left(P_S + P_K - \rho \dfrac{K}{\tau} \right) \\ Vol\left[\left(1 - C_{\varepsilon_1}\left[P_S + P_K \right] \dfrac{\tau}{K} + D_K - D_\tau + \left(C_{\varepsilon_2} f_2 - 1 \right) \rho \right] \right] \end{bmatrix} \tag{5.44}$$

$$P_K = -\frac{2}{3} \rho\, K \left[\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z} \right] \tag{5.45}$$

$$P_S = \left[2\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial z}\right)^2 \right] + \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right]^2 + \left[\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right]^2 + \left[\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right]^2 - \frac{2}{3}\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right]^2 \right] \tag{5.46}$$

$$D_K = \frac{2}{K}\mu_{\tau_1}\left(\frac{\partial K}{\partial x}\frac{\partial \tau}{\partial x} + \frac{\partial K}{\partial y}\frac{\partial \tau}{\partial y} + \frac{\partial K}{\partial z}\frac{\partial \tau}{\partial z} \right) \tag{5.47}$$

$$D_\tau = \frac{2}{\tau}\mu_{\tau_2}\left(\left(\frac{\partial \tau}{\partial x}\right)^2 + \left(\frac{\partial \tau}{\partial y}\right)^2 + \left(\frac{\partial \tau}{\partial z}\right)^2 \right) \tag{5.48}$$

with

$$\frac{\partial u}{\partial x} = \left(\frac{\partial u}{\partial \xi} S_i^x + \frac{\partial u}{\partial \eta} S_j^x + \frac{\partial u}{\partial \zeta} S_k^x \right) \frac{1}{vol} \tag{5.49}$$

$$\frac{\partial u}{\partial y} = \left(\frac{\partial u}{\partial \xi} S_i^y + \frac{\partial u}{\partial \eta} S_j^y + \frac{\partial u}{\partial \zeta} S_k^y \right) \frac{1}{vol} \tag{5.50}$$

$$\frac{\partial u}{\partial z} = \left(\frac{\partial u}{\partial \xi} S_i^z + \frac{\partial u}{\partial \eta} S_j^z + \frac{\partial u}{\partial \zeta} S_k^z \right) \frac{1}{vol} \tag{5.51}$$

etc.

$$\mu_K = \left(\mu_l + \lambda_K \mu_t\right)$$

$$c_{\varepsilon_1} = 1.44 \qquad c_\mu = 0.09$$

$$\mu_{\tau_1} = \mu_{\tau_2} = \mu_\tau \qquad \lambda_\tau = 0.7353 \qquad \lambda_K = 0.7353 \tag{5.52}$$

$$\mu_\tau = \left(\mu_l + \lambda_\tau \mu_t\right) \tag{5.53}$$

$$R_T = \frac{\rho K \tau}{\mu_l} \tag{5.54}$$

$$C_{\varepsilon_2} = 1.83\left(1 - \frac{2}{9}\exp\left(-\left(\frac{R_T}{6}\right)^2\right)\right) \tag{5.55}$$

$$f_2 = \left(1 - \exp\left(-\frac{y^+}{4.9}\right)\right)^2 \tag{5.56}$$

$$f_\mu = \left(1 + \frac{3.45}{\sqrt{R_T}}\right)\tanh\left(\frac{y^+}{70}\right) \tag{5.57}$$

$$y^+ = \left(\frac{y_n \upsilon_\tau \rho}{\mu_\kappa}\right) \tag{5.58}$$

$$\mu_t = \mu_l \, c_\mu \, f_\mu \, R_T \tag{5.59}$$

6. Numerical Algorithm

If one wants to solve a partial differential equation numerically one ultimately must discretize and hence reduce the partial differential equation to a system of algebraic equations. There are many possibilities of achieving this. We will not describe here all the methods which have been successfully applied in the past but will restrict ourselves to those methods which have been implemented in KAPPA.

In KAPPA we use the so called „semi-discrete method". This is a discretization process in two stages, first discretizing only in space with the finite-volume method, leaving the problem continous in time. This leads to a system of ordinary differential equations in time. We then discretize in time using either the explicit Runge-Kutta method or the implicit LU-SSOR method for systems of ordinary differential equations.

Explicit methods typically need less computational work and are simpler both in derivation and application. Implicit methods, although expensive in computation, have less severe stability bounds (classical stability analysis shows unconditional stability but in practice nonlinear problems bounds are encountered). The extra work required for an implicit scheme is usually compensated by the advantages obtained by the increased

stability limits, and in general implicit schemes have been useful and successful for a variety of inviscid and viscous flow-field calculations. On the other hand, the explicit methods, although restricted by the small time steps due to fine grid spacing for numerical resolution, have been made competitive by the use of convergence acceleration techniques.

With the advantage of high speed vector and parallel computers one must also consider the degree to which a certain algorithm can be vectorized resp. parallelized when choosing a scheme. As a rule explicit schemes are more easily vectorized resp. parallelized than implicit schemes.

Another consideration is the question of time accuracy versus non-time-accurate steady state iteration. For unsteady problems we wish to employ time accurate methods, initialize the flow with some realizable state and integrate forward in time whith time steps commensurate with the unsteady phenomena which are being calculated. Both implicit and explicit methods are capable of computing time accurately. In steady state calculation we wish to integrate from some arbitrary state to the asymptotic solution in a way which will get us there with the least amount of computational work. Non-time-accurate techniques e.g. relaxation methods, variable time steps, multigrid techniques can be employed as long as they are convergent and do not distort the steady state equations so as to produce inaccurate results.

6.1 Explicit method

Stable time stepping methods for the semi-discretized Navier-Stokes equations can be patterned on standard schemes for ordinary differential equations. Multistage schemes of the Runge-Kutta type have the advantage that they do not require any special starting procedure, in contrast to leap frog and Adams Bashforth methods, for example [46]. These schemes are usually designed to give a high order of accuracy. If the objective is simply to obtain a steady state as fast as possible, the order of accuracy is not important. This allows the use of schemes selected purely for their properties of stability and damping [4, 29]. If the cell volume V_{ijk} is independent of time, the semi-discretized N-S equations can be written as:

$$\frac{\partial w}{\partial t} + R_{(w)} = 0, \tag{6.1}$$

where $R_{(w)}$ is the residual.

Let w^n be the value of w after n time steps. The general m stage hybrid scheme to advance a time step Δt can be written as:

$$\begin{aligned} w^{(0)} &= w^{(n)} \\ w^{(1)} &= w^{(0)} - \alpha_1 \Delta t R^{(0)} \\ &\vdots \\ w^{(m-1)} &= w^{(0)} - \alpha_{m-1} \Delta t R^{(m-2)} \end{aligned}$$

$$w^{(m)} = w^{(0)} - \alpha_{m-1}\Delta t R^{(m-1)}$$
$$w^{(n+1)} = w^{(m)}$$

Where appropriate coefficients α_m can extend the stability region considerably [30].

6.2 Implicit method

An approximate LU decomposition method developed by Jameson and Turkel [45] applied in two dimensions by Jameson and Yoon [49] modified and extended to three dimensions by Rieger and Jameson [31] is implemented in KAPPA. Since the classical direct LU decomposition of the unfactored implicit operator is too expensive for multi-dimensional problems one has to resort to incomplete or approximate LU decomposition methods. In the latter concept a particular approximation to the unfactored implicit operator is chosen so that the desired LU-representation will directly result. Hence for any dimensions only two factors appear which additionally have the advantage to be easily invertible. Usually this objective can be achieved by an appropriate lower order analogue of the original operator. For triangular matrices inversion is done by simple forward and backward sweeps across the field. Hence, these methods resemble the Symmetric Succesive Over-Relaxation (SSOR) approach. The numerical method for the semi-discrete Navier-Stokes equation then reads:

$$\underline{R} = \Delta_1\left(\underline{E} - \underline{E}_v - \underline{E}_D\right) + \Delta_2\left(\underline{F} - \underline{F}_v - \underline{F}_D\right) + \Delta_3\left(\underline{G} - \underline{G}_v - \underline{G}_D\right), \tag{6.2}$$

where $\underline{E}$, $\underline{F}$, $\underline{G}$ represent the inviscid and viscous flux vectors into the general coordinate directions x_1, x_2, x_3, respectively. For control and nonlinear instabilities and central differences a scheme has to be provided by suitable dissipation operators which are indicated by lower indices D.

Then a Newton-iteration would read:

$$\left(\frac{\partial \underline{R}}{\partial \underline{q}}\right)^n \delta \underline{q}^n + \underline{R}^n = 0. \tag{6.3}$$

Here the upper index indicates the iteration count and δq^n is defined as $\delta q^n = q^{n+1} - q^n$. In general, the computation and inversion of the functional matrix $\partial \underline{R}/\partial \underline{q}$ is too costly , so that an approximate form has to be found such that the inversion is easy and stable. A choice which has been found beneficial resembles that of flux-vector splitting. Because the particular construction only affects the implicit operator a rather crude choice fulfills the requirements for diagonal dominance of the coefficient matrices.

For definition of the functional matrix some Jacobians of the different flux vectors are needed.

$$\underline{\underline{A}} \equiv \frac{\partial \underline{E}}{\partial \underline{q}} \qquad \underline{\underline{A}}_{\upsilon} \equiv \frac{\partial \underline{E}_{\upsilon}}{\partial \underline{q}} \qquad \underline{\underline{A}}_{D} \equiv \frac{\partial \underline{E}_{D}}{\partial \underline{q}} \tag{6.4}$$

$$\underline{\underline{B}} \equiv \frac{\partial \underline{F}}{\partial \underline{q}} \qquad \underline{\underline{B}}_{\upsilon} \equiv \frac{\partial \underline{F}_{\upsilon}}{\partial \underline{q}} \qquad \underline{\underline{B}}_{D} \equiv \frac{\partial \underline{F}_{D}}{\partial \underline{q}} \tag{6.5}$$

$$\underline{\underline{C}} \equiv \frac{\partial \underline{G}}{\partial \underline{q}} \qquad \underline{\underline{C}}_{\upsilon} \equiv \frac{\partial \underline{G}_{\upsilon}}{\partial \underline{q}} \qquad \underline{\underline{C}}_{D} \equiv \frac{\partial \underline{G}_{D}}{\partial \underline{q}} \tag{6.6}$$

Then an appropriate approximation to the functional matrix in 6.3 could be:

$$\left(\frac{\partial \underline{\tilde{R}}}{\partial \underline{q}} \right)^{n} = D_{\xi} \underline{\underline{A}} + D_{\eta} \underline{\underline{B}} + D_{\zeta} \underline{\underline{C}}. \tag{6.7}$$

The experience shows that in principle only the inviscid flux Jacobians have to be considered for definition of the implicit operator , also for viscous calculations. Although robustness and stability may be improved for severe problems by including into the implicit operator an approximation of the artificial dissipation operator and by parts the physical viscous flux Jacobians, for a further discussion the basic scheme [31] is sufficient.

The difference operator D_{ξ} etc. are written as a sum of first order forward (Δ_{ξ}) and backward (∇_{ξ}) difference operators:

$$D_{\xi} \underline{\underline{A}} = \Delta_{\xi} \underline{\underline{A}}^{-} + \nabla_{\xi} \underline{\underline{A}}^{+}. \tag{6.8}$$

Now, the particular flux Jacobians $\underline{\underline{A}}^{+}$ and $\underline{\underline{A}}^{-}$ etc. are defined in such a way that they possess only non-negative and non-positive eigenvalues:

$$\underline{\underline{A}}^{\pm} = \frac{1}{2} \left(\underline{\underline{A}} \pm r_{A} \underline{\underline{I}} \right), \tag{6.9}$$

$$r_{A} \geq \max\left(|\lambda_{A}| \right). \tag{6.10}$$

That is achieved by defining r_{A} as a value which has to be equal or greater than the spectral radius of $\underline{\underline{A}}$.

By sweeping forward and backward through the field the resulting relations can be combined similar to a SSOR method [32] and it turns out that the implicit operator in 6.7 can be approximately factorized into a product of a strictly lower triangular matrix

$\underline{\underline{L}}$, a diagional matrix $\underline{\underline{D}}$ and an upper triangular matrix $\underline{\underline{U}}$. Hence the basic scheme can be written as:

$$\left(\underline{\underline{L}}\,\underline{\underline{D}}^{-1}\,\underline{\underline{U}}\right)\delta\,\underline{q}^{n} = -\underline{R}^{n}, \tag{6.11}$$

with:

$$\underline{\underline{L}} = \nabla_{\xi}\underline{\underline{A}}^{+} + \nabla_{\eta}\underline{\underline{B}}^{+} + \nabla_{\zeta}\underline{\underline{C}}^{+} - \underline{\underline{A}}^{-} - \underline{\underline{B}}^{-} - \underline{\underline{C}}^{-}, \tag{6.12}$$

$$\underline{\underline{D}} = \left(r_A + r_B + r_C\right)\underline{\underline{I}}, \tag{6.13}$$

$$\underline{\underline{U}} = \Delta_{\xi}\underline{\underline{A}}^{-} + \Delta_{\eta}\underline{\underline{B}}^{-} + \Delta_{\zeta}\underline{\underline{C}}^{-} + \underline{\underline{A}}^{+} + \underline{\underline{B}}^{+} + \underline{\underline{C}}^{+}. \tag{6.14}$$

Now stability should be inhanced providing diagonal dominance for each factor [45]. Consider, as an example, the L-factor for which rearrangement will allow to write:

$$\begin{aligned}\underline{\underline{L}} = {} & \left(r_A + r_B + r_C\right)\underline{\underline{I}}_{i,j,k} - \frac{1}{2}\left(\underline{\underline{A}} + r_A\underline{\underline{I}}\right)_{i-1,j,k} \\ & - \frac{1}{2}\left(\underline{\underline{B}} + r_B\underline{\underline{I}}\right)_{i,j-1,k} - \frac{1}{2}\left(\underline{\underline{C}} + r_C\underline{\underline{I}}\right)_{i,j,k-1}.\end{aligned} \tag{6.15}$$

It is evident that diagonal dominance is only assured if the quantities $r_{Ai,j,k}$ and $r_{Ai-1,j,k}$ etc. are redefined so that they are equal and simultaneously the maximum of both original values defined after 6.10. That is:

$$r_{\overline{A}} = \max\left(r_{Ai,j,k}, r_{Ai-1,j,k}\right)_{orig}, \tag{6.16}$$

$$r_{Ai,j,k} := r_{A^+}, \qquad r_{Ai-1,j,k} := r_{A^-}. \tag{6.17}$$

Corresponding settings following for the U-factor suggest also a modified diagonal matrix D:

$$\underline{\underline{D}} = \frac{1}{2}\left(r_{A^-} + r_{A^+} + r_B + r_{B^+} + r_{C^-} + r_{C^+}\right)\underline{\underline{I}}. \tag{6.18}$$

In fact inversions of scheme 6.11 are accomplished by sweeping along diagonal planes I+J+K = const. across the domain. Then during the inversion process all variables needed from the off-diagonals are already updated, allowing a variation of the straightforward procedure. In inverting the modified L-factor we obtain from 6.11:

$$(r_A + r_B + r_C)\underline{\underline{I}}_{i,j,k}\delta\bar{\underline{q}}_{i,j,k} = -\underline{R}^n + \underline{\underline{A}}^+_{i-1,j,k}\delta\bar{\underline{q}}_{i-1,j,k} + \underline{\underline{B}}^+_{i,j-1,k}\delta\bar{\underline{q}}_{i,j-1,k} + \underline{\underline{C}}^+_{i,j,k-1}\delta\bar{\underline{q}}_{i,j,k-1}. \tag{6.19}$$

To avoid the explicit evaluation of Jacobian matrices the intermediate flux states can be approximated by a Taylor series expansion:

$$\tilde{\underline{E}}^+ = \left(\underline{E}^+\right)^n + \frac{\partial \underline{E}^+}{\partial \underline{q}}\left(\tilde{\underline{q}} - \underline{q}^n\right) + 0\left(\left|\delta\tilde{\underline{q}}\right|^2\right), \tag{6.20}$$

$$\delta\tilde{\underline{E}}^+ = \tilde{\underline{E}}^+ - \underline{E}^{+n} = \underline{\underline{A}}\delta\tilde{\underline{q}} + 0\left(\left|\delta\tilde{\underline{q}}\right|^2\right). \tag{6.21}$$

Now, the scheme 6.11 is inverted by the following steps:

$$\delta\tilde{\tilde{\underline{q}}}_{i,j,k} = -\underline{R}^n_{i,j,k} + \delta\tilde{\underline{E}}^+_{i-1,j,k} + \delta\tilde{\underline{F}}^+_{i,j-1,k} + \delta\tilde{\underline{G}}^+_{i,j,k-1}, \tag{6.22}$$

$$\delta\tilde{\underline{q}}_{i,j,k} = -\underline{D}^{-1}_{i,j,k}\delta\tilde{\tilde{\underline{q}}}_{i,j,k}, \tag{6.23}$$

$$\delta\underline{q}^n_{i,j,k} = \underline{D}^{-1}_{i,j,k}\left(\delta\tilde{\tilde{\underline{q}}}_{i,j,k} - \delta\underline{E}^-_{i+1,j,k} - \delta\underline{F}^-_{i,j+1,k} - \delta\underline{G}^-_{i,j,k+1}\right), \tag{6.24}$$

where:

$$\delta\tilde{\underline{E}}^{\pm} = \underline{E}^{\pm}\left(\tilde{\underline{q}}\right) - \underline{E}^{\pm}\left(\underline{q}^n\right), \tag{6.25}$$

$$\delta\tilde{\tilde{\underline{E}}}^{\pm} = \underline{E}^{\pm}\left(\tilde{\tilde{\underline{q}}}\right) - \underline{E}^{\pm}\left(\underline{q}^n\right). \tag{6.26}$$

In comparison to the straightforward inversion of scheme 6.11 no degradation in performance was observed with the cost-effective relaxation-type inversion 6.23.

6.3 Stability analysis

The numerical schemes for the solution of a partial differential equation must be stable. This means that any perturbation of the input values at the nth time level should be prevented from growing without bound. The stability of a scheme can be investigated by the von Neumann Analysis.

Consider a distribution of errors at any time in a mesh written as a series of the form:

$$\varepsilon(x,t) = \sum_m b_{m(t)} \cdot e^{ik_m x}. \tag{6.27}$$

If the amplification factor $g = \left| \frac{\varepsilon(x_0, t_0 + \Delta t)}{\varepsilon(x_0, t_0)} \right| < 1$ then the numerical scheme is stable. A similar technique can be used to investigate the stability for systems of equations [38]. A model for the equation encountered in fluid mechanics can usually be written in the form:

$$\frac{\partial W}{\partial t} + \frac{\partial F_{(w)}}{\partial x} = \frac{\partial W}{\partial t} + [A]\frac{\partial W}{\partial x} = 0. \tag{6.28}$$

We now locally linearize the system by holding [A] constant while the W vector is advanced through a single time step. For an explicit scheme this analysis leads to the requirement that $\left| \lambda_{\max} \frac{\Delta t}{\Delta x} \right| \leq 1$ where λ_{max} is the largest eigenvalue of the [A] matrix.

We apply this analysis to the K-ε model in generalized curvilinear coordinates. A conservative estimate for a nominal Courant number of unity is

$$\Delta t = \frac{Vol}{\sum_i \lambda_i}, \tag{6.29}$$

where λ_i are the averaged spectral radii of the Jacobian matrices in i, j and k directions. Linearizing around a reference state the K-ε equations in matrix form are:

$$\frac{\partial W}{\partial t} + A\frac{\partial W}{\partial \xi} + B\frac{\partial W}{\partial \eta} + C\frac{\partial W}{\partial \zeta} + D\frac{\partial^2 W}{\partial \xi^2} + E\frac{\partial^2 W}{\partial \eta^2} + F\frac{\partial^2 W}{\partial \zeta^2} = G \cdot W, \tag{6.30}$$

with:

$$W = Vol\begin{bmatrix} \rho K \\ \rho \varepsilon \end{bmatrix} \tag{6.31}$$

and:

$$A = \begin{bmatrix} U & 0 \\ 0 & U \end{bmatrix} \tag{6.32}$$

$$B = \begin{bmatrix} V & 0 \\ 0 & V \end{bmatrix} \tag{6.33}$$

$$C = \begin{bmatrix} W & 0 \\ 0 & W \end{bmatrix} \tag{6.34}$$

$$U = us_i^x + vs_i^y + ws_i^z \qquad V = us_j^x + vs_j^y + ws_j^z \qquad W = us_k^x + vs_k^y + ws_k^z \tag{6.35}$$

$$D = \begin{bmatrix} \dfrac{-\frac{\mu_k}{\rho}c_1 + c_4 + c_7}{Vol} & 0 \\ 0 & \dfrac{-\frac{\mu_\varepsilon}{\rho}c_1 + c_4 + c_7}{Vol} \end{bmatrix} \tag{6.36}$$

$$E = \begin{bmatrix} \dfrac{-\frac{\mu_k}{\rho}c_2 + c_5 + c_8}{Vol} & 0 \\ 0 & \dfrac{-\frac{\mu_\varepsilon}{\rho}c_2 + c_5 + c_8}{Vol} \end{bmatrix} \tag{6.37}$$

$$F = \begin{bmatrix} \dfrac{-\frac{\mu_k}{\rho}c_3 + c_6 + c_9}{Vol} & 0 \\ 0 & \dfrac{-\frac{\mu_\varepsilon}{\rho}c_3 + c_6 + c_9}{Vol} \end{bmatrix} \tag{6.38}$$

$$G = \begin{bmatrix} 0 & -c_3 Vol \\ -c_2' \frac{\varepsilon^2}{K^2} c_3 Vol & -2c_2' c_3 Vol \frac{\varepsilon}{K} \end{bmatrix}. \tag{6.39}$$

The eigenvalues are readily obtained by calculating $det(A - \lambda I) = 0$. The largest eigenvalue then reads:

$$\lambda_A = uS_i^x + vS_i^y + wS_i^z \tag{6.40}$$

$$\lambda_B = uS_i^x + vS_i^y + wS_i^z \tag{6.41}$$

$$\lambda_C = uS_i^x + vS_i^y + wS_i^z \tag{6.42}$$

$$\lambda_D = -\frac{\mu_k}{\rho}\left(\frac{c_1 + c_4 + c_7}{Vol}\right) \tag{6.43}$$

$$\lambda_E = -\frac{\mu_k}{\rho}\left(\frac{c_2 + c_5 + c_8}{Vol}\right) \tag{6.44}$$

$$\lambda_F = -\frac{\mu_k}{\rho}\left(\frac{c_3 + c_6 + c_9}{Vol}\right) \tag{6.45}$$

$$\lambda_G = c_\varepsilon Vol \frac{\varepsilon}{K}\left(c_2' + \sqrt{c_2'^2 - c_2'}\right) \tag{6.46}$$

and the time step can be calculated according to:

$$\Delta t = \frac{Vol}{\lambda_A + \lambda_B + \lambda_C + \lambda_D + \lambda_E + \lambda_F + \lambda_G}. \tag{6.47}$$

The estimations of the time step for the K-τ model and Navier-Stokes is similar and will not be repeated here (for reference see [20]).

6.4 Artificial dissipation

Whenever discrete methods are used to calculate complex 3D-flows including shocks, or to compute high Reynolds number behaviour, scales of motion appear which cannot be resolved by the numerics [26]. These can be brought about by the nonlinear interactions in the convection terms of the momentum equations (resp. two equation turbulence model). If a scale is represented by wavelength or frequency , it can be easily shown that two waves interact as products to form a wave of higher frequency (the sum of the original two) and one of lower frequency (the difference).

The lower frequencies do not cause a problem, but the continual cascading into higher and higher frequencies does. It is accounted for physically by shock formation or by viscous dissipation of the very high wave numbers. In numerical computations it cannot be ignored and must be accounted for in the algorithm constructed. In any finite discrete mesh the cascading frequencies can eventually exceed the capacity of the mesh resolution at which point they can either: a) alias back into the lower frequencies or b) pile up at the higher frequency side. In either case, if uncontrolled, these terms can lead to serious inaccuracies and possible numerical instability.

For the central difference type scheme used in KAPPA a numerical dissipation model is included in this scheme. The form of this dissipation model is a blending of second-difference and fourth-difference terms [46]. The second-difference terms are used to prevent oscillations at shock waves, while the fourth-difference terms are important for stability and convergence to a steady state. The dissipaton model used in KAPPA was first introduced by Jameson, Schmidt and Turkel [46] for the Euler equations. Several modifications of the model have been investigated in [47] and [30] in order to improve it and make it suitable to obtain accurate and efficient solutions of the Navier-Stokes equations.

Basically the same dissipation model has been used for the two equation turbulence models. We will describe these models here only for the K-ε model and refer the

interested reader to the report of Magagnato [13] and Leicher [24] for the Navier-Stokes equations in 3D.

The linearized K-ε equations in generalized curvilinear coordinates reads:

$$\frac{\partial W}{\partial t} + A\frac{\partial W}{\partial \xi} + B\frac{\partial W}{\partial \eta} + C\frac{\partial W}{\partial \zeta} + D\frac{\partial^2 W}{\partial \xi^2} + E\frac{\partial^2 W}{\partial \eta^2} + F\frac{\partial^2 W}{\partial \zeta a^2} = G \cdot W . \quad (6.48)$$

where the definitons of the matrices A...G and W can be found in the preceeding paragraph. To advance the scheme in time we use a multistage scheme. A typical step of a Runge-Kutta approximation to this equation is

$$W^{(m)} = W^{(0)} - \alpha_m \frac{\Delta t}{Vol}\Big[D_\xi A^{(m-1)} + D_\eta B^{(m-1)} + D_\zeta C^{(m-1)} + D_{\xi^2} D^{(m-1)} + D_{\eta^2} E^{(m-1)} + D_{\zeta^2} F^{(m-1)} - G^{(m-1)} - AD\Big] \quad (6.49)$$

where D_ξ, D_η, D_ζ, D_{ξ^2}, D_{η^2}, D_{ζ^2} are first resp. second spatial differene operators, and represents the artificial dissipation terms. The dissipation terms consist of fourth differences. That is:

$$AD = -(D_\xi^4 + D_\eta^4 + D_\zeta^4)W , \quad (6.50)$$

where

$$D_\xi^4 = \nabla_\xi[(\lambda_{i+1\,2,j,k}\kappa^{(4)})\Delta_\xi \nabla_\xi \Delta_\xi]W_{i,j,k} . \quad (6.51)$$

D_η^4 and D_ζ^4 are defined analogously. ∇_ξ and Δ_ξ are the standard forward and backward difference operator respectively associated with the ξ direction and $\kappa^{(4)}$ is a constant.

The variable scaling factor λ_ξ, λ_η and λ_ζ could be taken proportionally to the largest eigenvalues of the matrices A, B and C resp. if one would solve the K–ε model uncoupled with the Navier-Stokes equations. But in practice the Navier-Stokes and the K–ε model are solved simultaneously , so that the largest eigenvalue of the coupled N-S and K–ε equations must be taken. Since the eigenvalues of the N-S equations are larger than those from the K–ε model one must take the former.

They are:

$$\lambda_\xi = \left|uS_i^x + vS_i^y + wS_i^z\right| + c\sqrt{\left(S_i^x\right)^2 + \left(S_i^y\right)^2 + \left(S_i^z\right)^2} \quad (6.52)$$

$$\lambda_\eta = \left|uS_j^x + vS_j^y + wS_j^z\right| + c\sqrt{\left(S_j^x\right)^2 + \left(S_j^y\right)^2 + \left(S_j^z\right)^2} \quad (6.53)$$

$$\lambda_\zeta = \left|uS_k^x + vS_k^y + wS_k^z\right| + c\sqrt{\left(S_k^x\right)^2 + \left(S_k^y\right)^2 + \left(S_k^z\right)^2} \quad (6.54)$$

where c is the speed of sound.

For meshes with cell aspect ratios $0(10^3)$ typically generated for high Reynolds number viscous flows, Swanson and Turkel [47] and also Martinelli and Jameson [30] proposed to adjust the scaling factors. Instead of using the anisotropic formulation:

$$\lambda_{i+1/2,j,k} = \lambda_\xi \tag{6.55}$$

$$\lambda_{i,j+1/2,k} = \lambda_\eta \tag{6.56}$$

$$\lambda_{i,j,k+1/2} = \lambda_\zeta \tag{6.57}$$

they adjust the scaling factors as a function of the spectral radii of the Jacobian matrices associated with the ξ, η, ζ directions and account for varying cell aspect ratio

$$\lambda_{i+1/2,j,k} = \lambda_{i+1/2,j,k} \cdot \phi_{i+1/2,j,k} \tag{6.58}$$

$$\lambda_{i,j+1/2,k} = \lambda_{i,j+1/2,k} \cdot \phi_{i,j+1/2,k} \tag{6.59}$$

$$\lambda_{i,j,k+1/2} = \lambda_{i,j,k+1/2} \cdot \phi_{i,j,k+1/2} \tag{6.60}$$

where

$$\phi_{i+1/2,j,k} = 1 + max\left(\left(\lambda_\eta / \lambda_\xi\right)^\alpha \cdot \left(\lambda_\zeta / \lambda_\xi\right)^\alpha\right) \tag{6.61}$$

and analoguously for $\phi_{i,j+1/2,k}$ and $\phi_{i,j,k+1/2}$.

7. Convergence Acceleration Technique

For the calculation of stationary flows it is interesting to accelerate the convergence to steady state as quickly as possible. Several techniques are applied in KAPPA in order to minimize the computational effort. The convergence acceleration technique cannot be used for all possible combinations of numerical schemes and/or physical equations. For example, the local time stepping technique is senseless if one adopts the implicit LU-SSOR scheme or the enthalpy damping technique for a Navier-Stokes calculation. Anyway, the code switches off automatically those accelerating techniques which are not appropriate for the specific computation.

7.1 Local time stepping

For explicit numerical schemes the time step is limited by the Courant Friedrich Lewy condition (CFL), which requires that the domain of dependence of the numerical scheme must at least contain the region of dependence of the original differential equation. That means that the time step is a function of the cell size. The ratios between the smallest and the biggest cell size are of the order of $0(10^8)$ for viscous flow meshes. Computation using the time step of the smallest cell in the whole computational domain would be very costly . The concept of local time stepping overcomes this difficulty by advancing every computational cell at its own stability limit. A conserva-

tive estimate of the local time step limit for the Euler equation is:

$$\Delta t = \frac{Vol \ln \alpha}{\lambda_\xi + \lambda_\eta + \lambda_\zeta}, \tag{7.1}$$

where λ_ξ, λ_η and λ_ζ are estimates of the local maximum wave speed in the ξ, η and ζ-direction of the convective operator. For the Navier-Stokes equations the contribution due to the diffusive operator has to be added:

$$\Delta t = \frac{Vol}{(\lambda_\xi + \lambda_\eta + \lambda_\zeta) + (\upsilon_\xi + \upsilon_\eta + \upsilon_\zeta)}. \tag{7.2}$$

And finally the two equation turbulence model has a time step limit which is augmented by an estimate of the source terms:

$$\Delta t = \frac{Vol}{(\lambda_\xi + \lambda_\eta + \lambda_\zeta) + (\upsilon_\xi + \upsilon_\eta + \upsilon_\zeta) + \mu_s}. \tag{7.3}$$

7.2 Implicit residual averaging

Another technique to enhance the convergence rate of an explicit scheme is the implicit residual averaging. For a multistage scheme the residuals in each Runge-Kutta step are replaced by a weighted average of the residuals at the neighbouring point. For three-dimensional flows the implicit residual smoothing is applied in the product form:

$$(1 - \varepsilon_\xi \nabla_\xi \Delta_\xi)(1 - \varepsilon_\eta \nabla_\eta \Delta_\eta)(1 - \varepsilon_\zeta \nabla_\zeta \Delta_\zeta)\overline{R}_{ijk} = R_m, \tag{7.4}$$

where: R_m – is the explicit residual at the point i, j, k and $\overline{R}_{ijk}$ – is the smoothed residual. For highly stretched cells Martinelli [20] has given a formula in which the smoothing coefficient is a function of the characteristic wave speed. In three-dimensions this can be de fined as [22]:

$$\varepsilon_\xi = max\left(\left(\frac{1}{4}\frac{CFL}{CFL_{ex}}\frac{\lambda_\xi}{\lambda_\xi + max(\lambda_\eta \cdot \lambda_\zeta)}\left(1 + max\left(\left(\frac{\lambda_\eta}{\lambda_\xi}\right)^\alpha \cdot \left(\frac{\lambda_\zeta}{\lambda_\xi}\right)^\alpha\right)\right)^2 - 1\right) \cdot C\right), \tag{7.5}$$

and analoguously the η and ζ coefficient. In KAPPA we replace the lower limit $c = 0$, proposed by Martinelli [20] by $c = 0.1$ for the Navier-Stokes equations and $c = 0.2$ for the two equation model.

7.3 Multigrid

A very efficient convergence acceleration technique is the multigrid strategy. Although proposed in 1964 by Federenko [23] it has been applied only recently for

viscous calculations by Martinelli [20]. The general idea behind any multigrid is to transfer some of the task of tracking the evolution of the system to a sequence of successively coarser meshes [5]. This has two advantages. First, the computational effort per time step is reduced on a coarser mesh. Second, the use of larger control volumes on the coarser grids tracks the evolution on a larger scale, with the consequence that global equilibrium can be more rapidly attained. The coarser meshes are generated by elimination of every other line in the i, j and k-direction. Sweeping through the coarser meshes in different ways influences the efficiency and the damping property of the numerical scheme. In KAPPA the V-cycle can be used to accelerate the convergence. In most cases the multigrid strategy increases the convergence rate by a factor of 5 to 20 by increasing the computational work by a factor of 2 for a multigrid cycle, so that the cpu-time can be reduced effectively by a factor between 2 and 10 using the multigrid technique.

The principle concept of the multigrid is to combine the coarse grid efficiency with the fine grid accuracy. Analytically, error is the sum of infinite number of frequencies in a Fourier series representation. On a given grid the visible modes of the error are finite and depend on the grid spacing. A short wavelength (high frequency) wave on a fine grid can be represented as a long wavelength (low frequency) wave on a coarser grid, (the so-called aliasing effect).

Low frequency components of error are badly damped by the conventional iterative techniques employed in the solution methods, as opposed to high frequency modes which can be damped very effectively. The mathematical basis underlying the multigrid method is to damp different error modes on different grids effectively by making use of the aliasing principle. A practical conclusion of this is that it is advantageous to use as many grids as possible to cover a large spectrum.

The following steps are performed in the multigrid computations:

- *Smooth* the error on the fine grid,
- transfer this *solution* and the *residuals* to a coarser grid (*restriction*), and solve on the coarse grid,
- transfer back the *corrections*, the difference between the transferred solution from the finer grid and the solution obtained on the coarser grid, to the finer grid by interpolation (*prolongation*).

Then the new solution on the fine grid is obtained by adding these coarse grid corrections to the fine grid solution, Figure 7.1 (a). This one cycle is called the

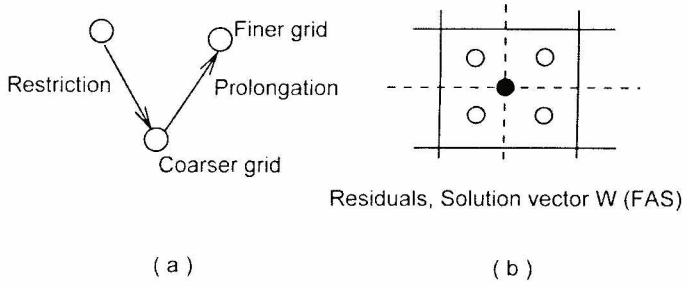


Figure 7.1 *(a) Multigrid operations for a V-cycle, (b). Grid coarsening.*

V-cycle. It is remarked here that the reason that the residuals are also restricted to the coarser grids is that the solution on the coarser grids is driven by the residuals on the finest grid, to ensure converged solution on the coarser grids once converged solution on the finest grid is obtained. This is done by introducing a so-called forcing function which is the difference between the residual on the coarser grid and the restricted residual from the finer grid. This multigrid algorithm where the solution and the residuals are used is called *Full Approximation Storage* (*FAS*) *scheme.*

A coarse grid is obtained by deleting every other coordinate line in each direction, as shown in Figure 7.1 (b). In practice, in addition to the multigrid operators *restriction* and *prolongation*, additional *smoothing* is performed to ensure a smooth representation of the errors, (elimination of the high frequency components of the error is essential to the success of the multigrid method).

Multigrid operations begin on the finer grid. However, iterations beginning from the initial flow field, commonly initialized with the freestream flow values, may deteriorate or even preclude convergence on the finest grid. The solution to this problem is the so-called *Full Multigrid* (*FMG*) technique, where converged solutions on the intermediate grids are used as the initial solution on the finest grid as illustrated in Figure 7.2. In this example it is assumed that 5 grids are available in each block, the finest grid is denoted by h and coarser grids are denoted by $2h$, $3h$ etc. Multigrid operations begin on the grids $3h$, $4h$ and $5h$. A user specified **N1** number of V-cycles are performed on these grids to obtain a converged solution on grid $3h$ (here **N1** is used to indicate an input, in ***KAPPA*** the number of V-cycles for each level is element of an array, **N1** is not the name of the variable). In the second level, the solution on grid $3h$ is interpolated to the finer grid denoted by $2h$ and **N2** number of V-cycles are performed on these 4 grids. Converged solution on grid $2h$ after **N2** iterations is interpolated to the finest grid and final **N3** number of V-cycles are performed on the complete 5-grid system to obtain the final converged solution on the finest grid h.

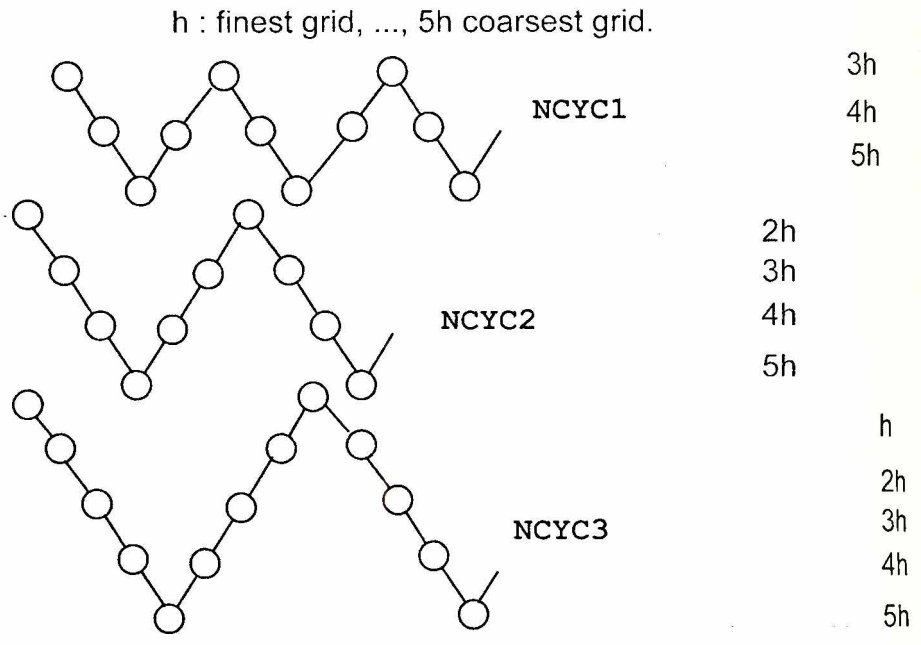


***Figure** 7.2 Full multigrid strategy.*

It is seen that there are three different iterations performed in the code:

1. Loop over multigrid levels, 3 in the example
2. loop over grids within a level, `N1`, `N2` *and* `N3` *in the example,*
3. loop on a given grid, typically 1 or 2.

Understanding of these iterations is essential for the understanding of the global structure of the code.

8. Boundary Conditions

The efficiency and generality of a flow code in terms of its ability to handle a wide variety of problems and flow conditions is highly dependent on the accuracy and the flexibility of the boundary conditions incorporated into the code. KAPPA possesses a number of different boundary conditions which can be selected to define the physical problem to be considered. These are the wall conditions, the moving wall condition, the symmetry condition, the farfield condition, the specification of a velocity profile or the static pressure etc. For block faces which coincide into a singular line or into a singular point there are a couple of boundary conditions allowing to describe the physics appropriately. Last but not least, the boundary of these blocks which adjoin another block face must be specified.

8.1 Solid wall and moving wall

The condition of no slip for viscous flow is imposed by setting the flow velocity to zero at solid surfaces or to specify the velocity of the moving wall at the surfaces. For inviscid calculations one must assume that the flow is tangential to the solid surface and the pressure is determined from the normal momentum equation.

8.2 Farfield condition

A very attractive boundary condition for the farfield should damp all outgoing disturbances effectively without falsifying the flow field too much. Therefor this condition is based on the introduction of the Riemann invariants for a one dimensional flow normal to the boundary. At the out flow boundary, the tangential velocity components and the entropy are extrapolated from the interior, while at the inflow boundaries they are specified with the free stream values. These two quantities together with the sum and the difference of the two invariants provide a complete definition of the flow in the farfield.

For the specification of the boundary condition for the two-equation turbulence model two parameters of the turbulent flow field need to be known. The first is known in most cases or at least can be guessed, namely the turbulence level Tu_∞. The other parameter, the turbulence length scale, is difficult to measure and in most cases not known a priori. One possible way to calculate the length scale L_t is to choose the eddy viscosity in the order of the laminar viscosity, and by applying the definition of the eddy viscosity $\mu_t = \rho c_\mu \sqrt{K} L_T$ the corresponding turbulence length scale can be calculated. A good compromise for the eddy viscosity is to scale it with the turbulence level, for example: $\mu_t \approx 1 + 2 \cdot Tu^2$ [14]. The two-equation models have different boundary conditions than those for the Navier-Stokes equations. At in flow boundaries both values of the turbulence model must be specified by Dirichlet boundary conditions while at out flow boundaries these are extrapolated from the interior. Also at the wall the boundary condition must be specified by a Dirichlet boundary condition, which sometimes is troublesome for the ε-equations. However, with the use of the K-τ model this reduces to $K = \tau = 0$ at the wall, which is computationally the best boundary condition for the wall.

9. Grid Generation

There is a wide variety of methods for generating grid systems. Algebraic methods such as conformal mapping, quadratic functions, or the control function approach of Eisemann [21] have been widely employed. The numerical approach of using elliptic solvers, Thompson, Thames and Mastin [35], is also widely used. Thompson [16] provides a good review of the current state of the art in grid generation. With increasing complexity of the geometry these methods become more and more in flexible, so that new concepts which should overcome these difficulties were developed. The subdivision of the entire volume network into adjacent sub-areas provides the possibil-

ity to arrange these sub-domains according to the actual geometry and thus to build up more complex grid structures at all. This approach is well known under the term of „block-structured grids". For the price of an additional necessary block-topology, increased flexibility in the possibilities of the geometry description are attained. Certainly the introduction of the block structure leads to a visible progress with respect to the treatable geometries, and based on this approach some almost automatic grid generation methods running in batch-mode have been well established for standard geometry configurations. But especially if the basic types of configurations are changing frequently, other weaknesses of this approach become evident. As all the steps of grid-generation and -control are carried out by means of batch programs, the editing of a base geometry, i.e. the evaluation of the geometry input for a mesh generator might lead to substantial difficulties. The actual generation programs require large lists of control parameters to determine the approach in advance. A visual inspection of the networks is only possible using plots and to do improvements implies again the use of the generation procedure. Due to the development of hard and software within the CAD/CAM area or in computer graphics in general during the last years, it becomes feasible to rearrange the entire preprocessing into graphic-interactive programs [52]. The advantages are obvious: within a dialog and under permanent visual control step by step (and even backwards) a basic geometry can be upgraded to a final network within one session. Errors can be cancelled immediately since they are easy to recognize, and possible variations can be tried at the smallest expanditure of time.

In our institute we are using the commercial software package ICEM-CFD for the generation of block structured grids.

10. Multiblock Topology

Conceptually multiblock topology is simple: the discretized flow domain, the grid, is divided into a number of blocks. From the parallel computation point of view, multiblock topology is a systematic way of distributing the domain among a number of processors in a non-overlapping manner. Even for a serial computation, however, multiblock topology is required for mesh generation about complex geometries due to practical difficulties in generating single block mesh. Multiblock topology is also useful in cases where the whole mesh cannot be stored in the core memory and only a number of blocks have to be solved at a time.

In the example shown in Figure 10.1, the flow domain is divided into 8 blocks of which the first 4 blocks, blocks 1, 2, 3 and 4 are assigned to processor number one and the other 4 blocks, blocks 5, 6, 7 and 8 are assigned to processor number two.

The structure of the code is not determined by the requirements of parallelization, it is the multiblock topology that determines the code structure. As shown in Figure 10.1, each processor has a local count of the blocks that are assigned to it and all parallel block loops run to that local number of blocks. For instance, in the example shown, a serial loop over blocks would be performed from 1 to 8, whereas in case of two processors, loops from 1 to 4 would be performed by the first processor, and from 5 to 8 by the second.

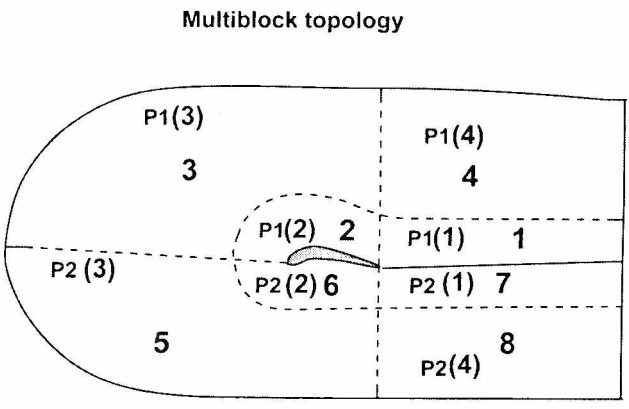


Figure 10.1 Multiblock topology example.

A block can have either a connection boundary, that is, it is connected to another block, or a physical boundary. In Figure 10.1, the block number 1 is connected to blocks 2, 4 and 7, and it has one physical boundary. Whether a data exchange between two blocks at the connection boundary is performed across processors or within a processor, makes no principle difference except for practical coding complications where a different routine has to be called for either case.

In all these cases a common requirement is that each block be individual, independent item of computation except for the global data exchange with other blocks at the connection boundaries. This is achieved in ***KAPPA*** by providing a halo of two-layers of dummy cells around each block.

A one-dimensional example of block connections and array dimensioning is shown in Figure 10.2. The number of grid points in i-direction is **NI**, in j-direction **NJ**, and in k-direction **NK**, as used in the code. In this example, the grid arrays are dimensioned

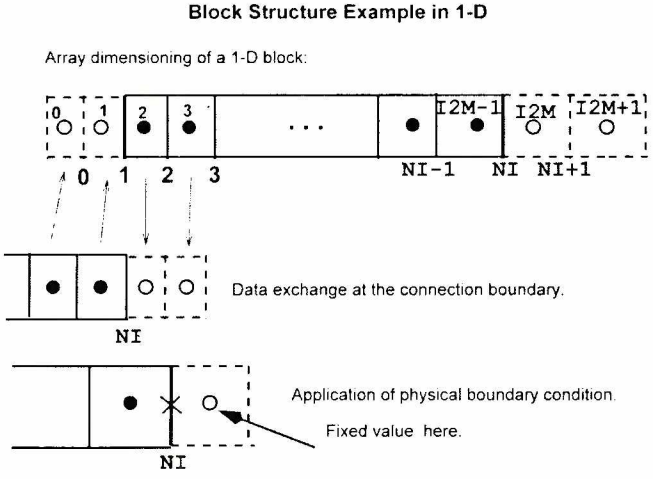


Figure 10.2 Block structure example in 1-D.

from `(0:NI+1)`. The first layer dummy cell coordinates are computed either by extrapolation from inner points for physical boundaries, or they are assigned the coordinates of the corresponding cells in the connected block for the connection boundaries. The flow variables are stored at the *cell-centers*, and flow variable arrays are dimensioned from `0` to `I2M+1` in i-direction.

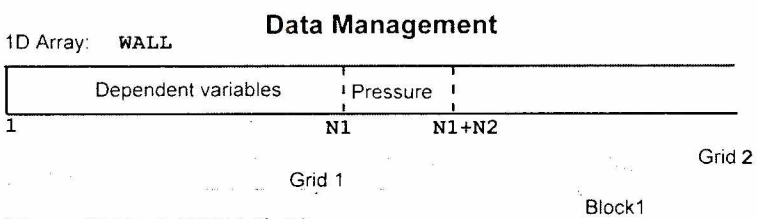
The cell-centers denoted by solid circles in the Figure indicate the internal flow field points which are computed during an iteration (for instance during a *Runge-Kutta stage*). Dummy cell values at the connection boundaries are assigned after the internal flow field is computed, by assigning the corresponding cell values of the connecting blocks, as indicated by arrows in Figure 10.2. This procedure is called *data exchange*. After the data exchange, the physical boundary conditions are applied by fixing the dummy cells so as to have the correct boundary value *at the boundary plane*. For instance, if the boundary value of W at the plane `NI` is W_{NI}, and the value of W at `I2M-1` is W_{I2M-1}, the value of the first dummy cell, W_{I2M}, is fixed as:

$$W_{I2M} = 2W_{NI} - W_{I2M-1},$$

so, that a linear interpolation between the first inner and dummy cell values give the correct value at the boundary. To fix the second layer of the dummy cell, either a higher order interpolation, or simple extrapolation is used.

Each block becomes a computationally independent unit once the dummy cell values are fixed.

11. Data Management




```
N1   = NVAR * NCELL(1,1)
N2   = NCELL(1,1)

ALLOC: IWALLP(IGRID,IB,1) = 1
       IWALLP(IGRID,IB,2) = N1+1

KAPPA: CALL RUNGEK(WALL)

RUNGEK: DIMENSION WALL(*)
        IBW= IWALLP(IGRID,IB,1)
        IBP= IWALLP(IGRID,IB,2)
        CALL CALCP(I2,J2,K2,WALL(IBW),WALL(IBP))

CALCP:  CALCP(I2M,J2M,K2M,W,P)
        DIMENSION W(0:I2M+1,0:J2M+1,0:K2M+1,NVAR)
        DIMENSION P(0:I2M+1,0:J2M+1,0:K2M+1)
```

Figure 11.1 Data management

A practical implication of the multigrid operations described above is the necessity of a flexible data management. In ***KAPPA*** a standard solution of using a large one dimensional work array **WALL** is chosen, Figure 11.1. The work array **WALL** is created by subroutine **ALLOCATEWALL** within each processor for the blocks assigned to that processor. For instance, in the example shown in Figure 10.1 the data for blocks 1, 2, 3 and 4 will be stored in processor 1, and data related to blocks 5, 6, 7 and 8 will be stored in processor 2, *blocks being numbered from 1 to the number of blocks assigned to that processor, i.e., numbered locally*. For each block, the data is stored *for each grid in that block, beginning from the finest grid of that level*. For each grid the same arrays are stored consecutively beginning with the dependent variables as indicated in Figure 11.1. These arrays are clearly indicated in subroutine **ALLOCATEWALL**. At the beginning of each new multigrid level, the data structure related to that level is created. In the example shown in Figure 7.2, at the third level, 5 grids will be stored for each block, the grid number 1 being the finest grid.

A *pointer* of an array is the *location of the first element of the array*. The allocation of the data structure is done by storing the pointers of each array for each grid and for each block. In the example shown in Figure 11.1, **IWALLP** is the pointer array for grid number **IGRID** of block **IB**. The first component of **IWALLP** points to the dependent variables, number of variables **NVAR** times the number of cells of that particular grid, grid 1 of block 1, indicated by **NCELL(1,1)**. Similarly, the second component is the pointer to the pressure array.

The subroutine **RUNGEKUTTA**, the driver routine for the solution using the Runge-Kutta method, is called from the main routine by only passing the **WALL** array, (Figure 11.1). Relevant data for the routines called from **RUNGEKUTTA** are passed to these routines by passing the *first element* of the arrays pointed by the pointer and their dimensions. In this example shown, the routine **CALCP** is called from **RUNGEKUTTA** by passing the dimensions **I2**, **J2**, and **K2** of the arrays **W** and **P**, and their first elements **WALL(IBW)** and **WALL(IBP)**, respectively. Conventional three dimensional representation of these arrays is recovered in the subroutine **CALCP**, (see also Figure 10.2 for the dimensions).

There are two main advantages of this data management structure:

1. Only the work array WALL has to be dimensioned as it is the only array seen by routines at the first and second levels of the calling-tree,
2. a complete generality is achieved in calling lower level routines, for instance the routine CALCP in the example given, the W and P arrays can belong to any grid of any block.

12. Global Structure of KAPPA

Once the items discussed above are clarified the global structure of ***KAPPA*** can be grasped easily. A block diagram of ***KAPPA*** code is shown in Figure 12.1. Input data

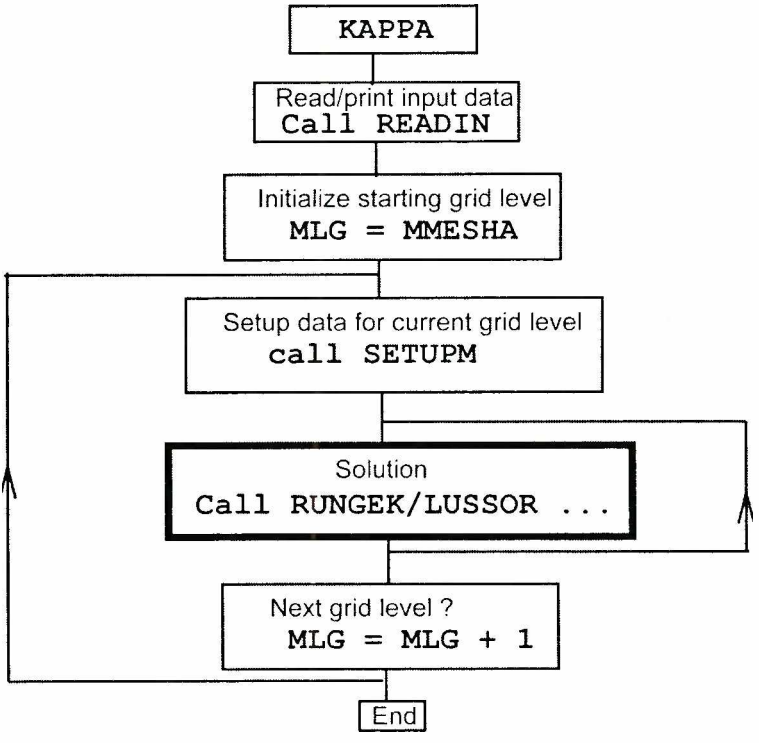


Figure 12.1 *Block diagram of KAPPA code*

is read and printed out in subroutine READIN which is called once. Then the starting grid level of the FMG is assigned to MLG. The loop on the multigrid levels MLG is the outmost iteration loop described in Section 4. For this grid level, the data setup is performed by the routine SETUPM which calls routines to read the mesh and boundary conditions, initial solutions or profiles. The work array WALL for this level is also created in SETUPM. Then the second loop over the number of V-cycles, N1 in Figure 7.2 is performed. In this loop the driving solver routine is called to do each V-cycle. This solver routine is either RUNGEKUTTA or LUSSOR as chosen by the user depending on whether the Runge-Kutta explicit or LU-SSOR implicit method is selected. Once these cycles are done, it is checked whether there is another level. The process is repeated for all the levels, for instance 3 in the example given in Figure 7.2 and the program stops.

Out of the three iteration loops described in Section 3, two loops are performed in the main routine KAPPA: loops over the FMG levels and the number of V-cycles within each level. The driver routine of the solution, depending on the numerical method

chosen, is either RUNGEKUTTA or LUSSOR. These solver routines are almost identical in structure as it will be discussed later in the report. Consequently, only the solver routine RUNGEKUTTA will be looked at in more detail.

A detailed block diagram of RUNGEKUTTA routine is shown in Figure 12.2. A single V-cycle is performed by the solver routine RUNGEKUTTA. This subroutine consists of two main parts:

1. First leg of the V-cycle where the grids from finest of that level to the coarsest are worked through in descending order,
2. the second leg of the V-cycle where grids beginning from the second coarsest to the finest are taken up in the ascending order.

The first part is performed in the routine in a linear fashion within the loop over grids from finest to the coarsest grid as shown in Figure 12.2. The second part is performed in subroutine RKPRO.

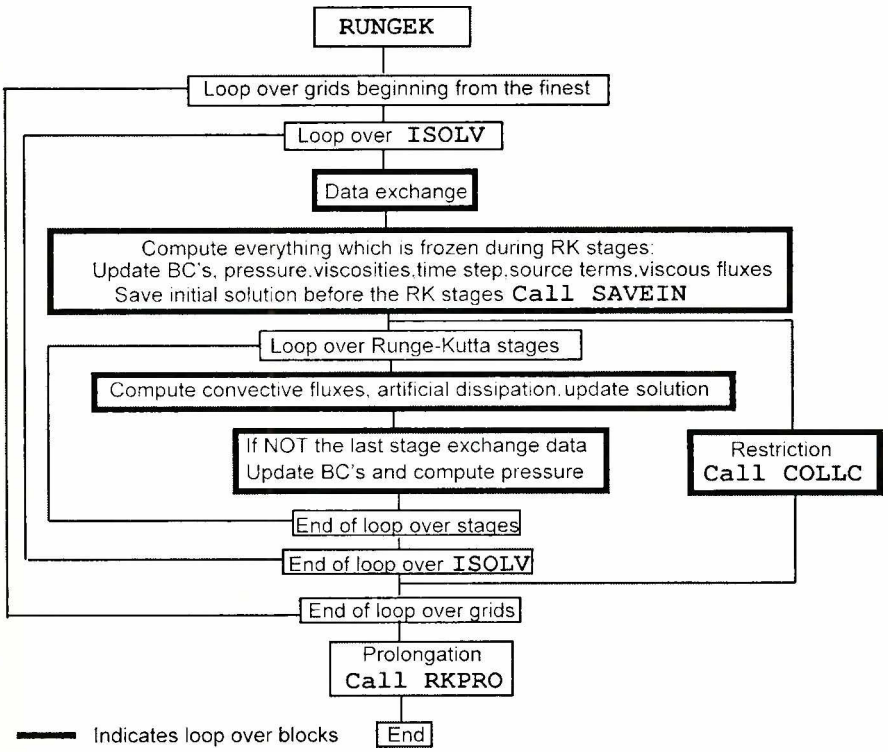


Figure 12.2 *Block diagram of RUNGEKUTTA subroutine*

Beginning with the finest grid, on each grid ISOLV number of iterations are performed. Inside these two loops data exchange is performed over each block, as described in Section 2. The bold face rectangles in Figure 12.2 indicate loops over blocks which are performed in parallel. After this operation the update of the boundaries is

performed by applying the boundary conditions (see Section 2). Everything which is frozen during the Runge-Kutta stages is computed before entering the loop over the Runge-Kutta stages: physical and eddy viscosities, time step, source terms and viscous fluxes. It is noted here that the source terms and viscous fluxes are only computed on the finest grid of the level. The initial solution (the current value of the dependent variable array W) is saved as required by the Runge-Kutta method in the routine SAVEIN. Within the following loop over the Runge-Kutta stages routine operations are performed: computation of the convective fluxes, artificial dissipations and solution update. Between the stages data exchange, boundary condition application and pressure computations are performed except for the last stage. Once loops over stages and ISOLV are finished, the solution is restricted within the subroutine COLLC (naturally, there is no restriction on the coarsest grid of the level) and the next coarser grid is selected in the loop over the grids, and the same operations are repeated. Once the coarsest grid is reached, the routine to perform the same operations from the second coarsest grid to the finest grid is called: RKPRO. Precisely the same operations as those described above are performed in the subroutine RKPRO with two differences: the outmost loop is over grids from the second coarsest grid to the finest, and prolongation routine ADDX is called instead of the restriction routine COLLC. For the implicit LU-SSOR method the driver routine LUSSOR is called instead of RUNGEKUTTA, which in turn calls the routine LUPRO instead of RKPRO. The simplicity of the global structure of KAPPA program is best illustrated in Figure 12.3 where the program

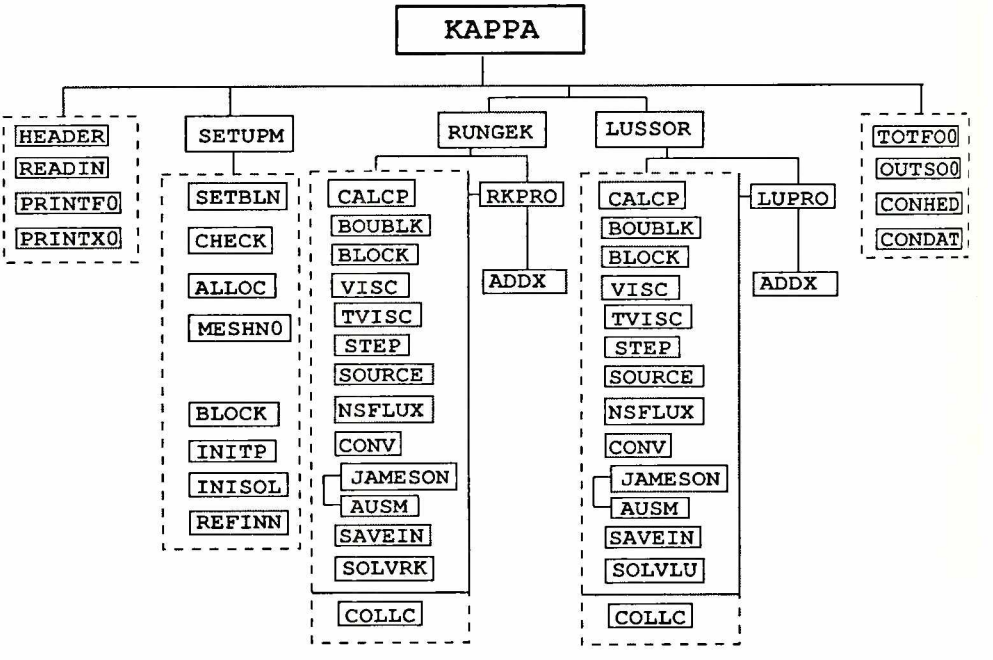


Figure 12.3 *Global calling-tree of program KAPPA*

calling-tree up to the third-level routines is depicted. The first and the last block of routines HEADER, READIN, PRINTF0, PRINTX0, TOTFO0, etc, are called once. The most important routine to understand and study among these routines is the data input routine READIN.

The second block routines headed by SETUPM is called for each level once. The routines SETBLN, where the block topology and boundary condition data is input, and ALLOCATEWALL, where the data management is performed by creating the work space WALL are the most crucial routines. The array bounds are computed in subroutine BLOCK which is also called from solver routines.

The driving routines for the solution are RUNGEKUTTA or LUSSOR. The first block under the routine RUNGEKUTTA headed by CALCP performs the operations described above and the routine names are self evident. For the convective fluxes, either central or upwind differencing can be employed calling JAMESON or AUSM, respectively. The Runge-Kutta multi-stage operations are performed in routine SOLVRK. The restriction routine COLLC is called on the way down from the finest grid as described above. The same block denoted with a solid line is also called from RKPRO, which calls the prolongation routine ADDX. The implicit method solver is LUSSOR from which the same routines in the third block are called with the difference that the LUSSOR logic is performed in SOLVLU as opposed to SOLVRK, and the loop over the grids on the way up is performed in LUPRO as opposed to RKPRO for the Runge-Kutta method. This repetition of the routines clearly facilitates the understanding and study of the code. For instance, both for implicit and explicit time integration schemes the right-hand side is computed using the same routines as shown in Figure 12.3. The left-hand side operations (the time integration) are either performed by SOLVRK or SOLVLU. The multigrid operations restriction and prolongation are performed by COLLC and ADDX, respectively. Critical variables that control the flow of the multigrid logic are MODE and KODE which are described in routines RUNGEKUTTA and LUSSOR and must be studied. A very important ingredient of the multigrid method is the smoothing, as explained in Section 3. These smoothing routines PSMOO and PSMOOC, which are called from SOLVRK and ADDX, respectively, are not shown in the Figure. Once this global structure presented here is clearly understood, study of any lower level routines which are below SETUPM, RUNGEKUTTA and LUSSOR or those which are not shown in Figure 12.3 is the study of implementation rather than the study of the structure of the code.

Finally, the detailed flowchart of KAPPA is given in Figure 12.4 where the global elements of program, the I/O routines, FMG control logic and convergence checks are shown.

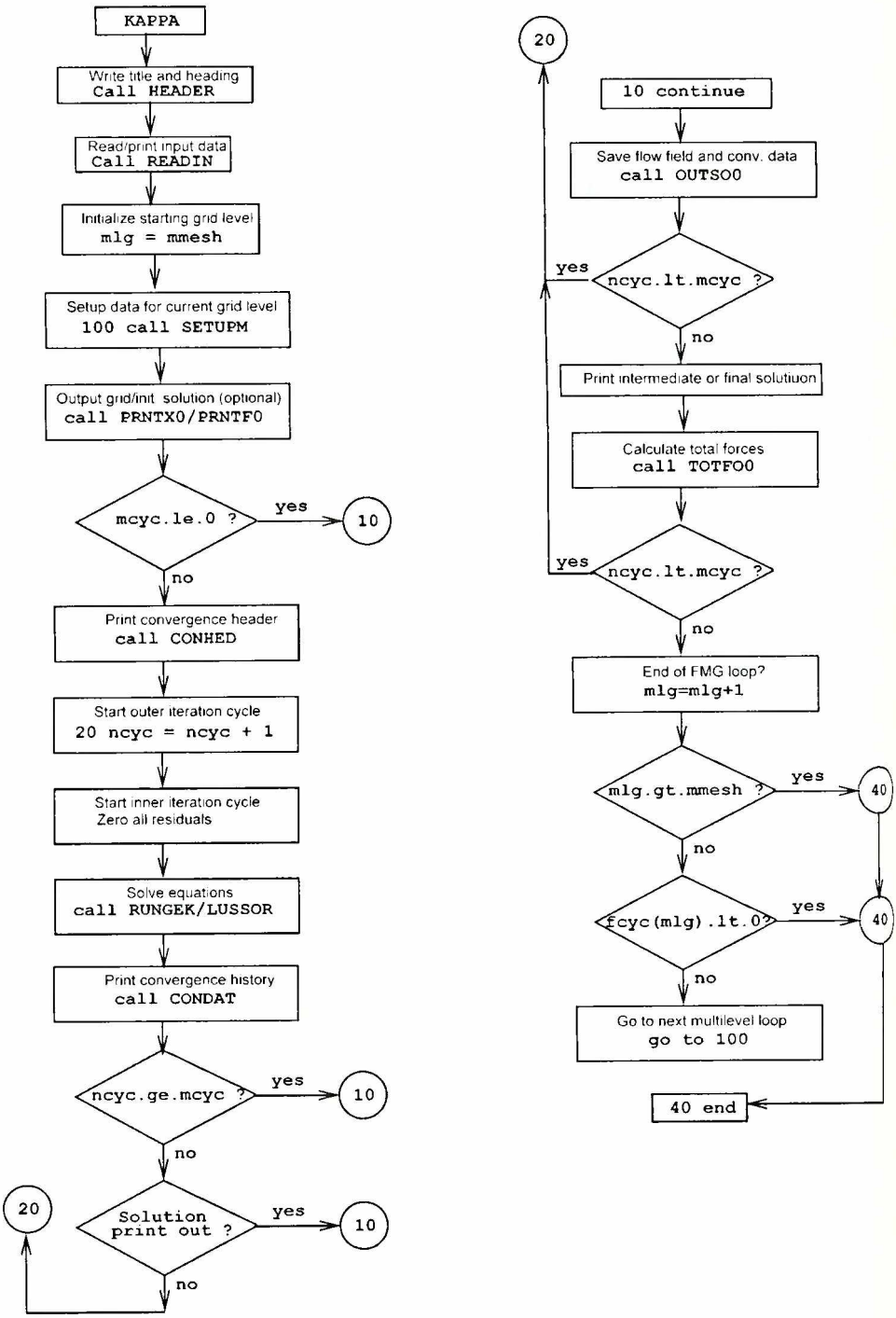


Figure 12.4 Flowchart of KAPPA

13. Applications

Since the code is based on the compressible Navier-Stokes equations it is in principle possible to calculate subsonic, transonic, supersonic and hypersonic flows. This requires that the code can handle the large variety in the mathematical nature of the governing equations. Needless to say, this is a very ambitious task and can hardly be fulfilled by a single numerical scheme. For example, supersonic flow requires the use of an implicit solver in order to solve the large disparity of eigenvalues in this flow regime while an explicit solver in conjunction with the multigrid strategy is best for low subsonic flows. Various two-dimensional and three-dimensional, internal and external, steady as well as unsteady flow fields have been calculated in the past in order to assess the accuracy and reliability of the code KAPPA. Some of these flows are discribed in this chapter.

13.1 Flow around ONERA M6 wing

The flow around the ONERA M6 wing is a standard test case in the CFD community since it is a very accurately measured and validated test for 3D transonic flows [1]. In order to capture the essential features of the flow a mesh consisting of about 760.000 points has been used, subdivided into 30 blocks for parallel computation (see figure 13.1). The freestream Mach number is Ma=0.84 the angle of attack $\alpha = 3.06°$ and the Reynolds number based on the mean chord length is $Re=11.7*10^6$.

The calculation has been made using the nonlinear two equation turbulence model of Craft et al. [11] without explicit forcing of the transition region. A sequence of four multigrid levels has been used to accelerate the convergence to the steady state solution. Appr. 300 iterations were necessary to achieve a converged solution. The contours of const. pressure coefficient are shown in Figure 13.2 demonstrating the two shocks forming at the leading edge and middle part of the wing. These two shocks merge into a single shock at the wing tip. The comparisons of the experimental pressure distribution with the calculation at y/s=0.2, y/s=0.65 and y/s=0.95 wing section are displayed in figures 13.3, 13.4, 13.5. The agreement are generally very good.

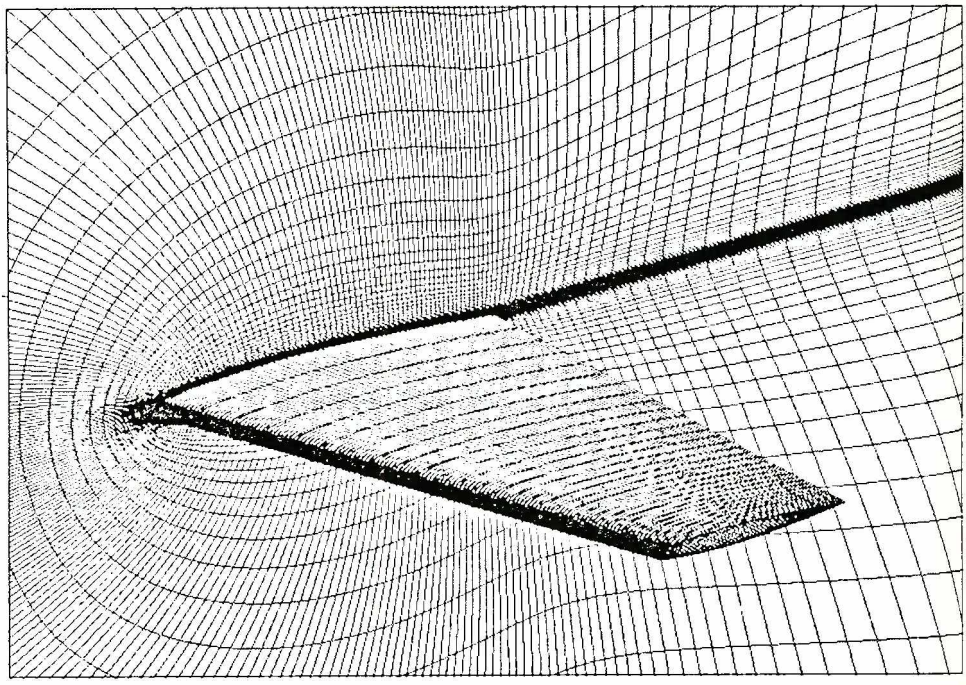


Figure 13.1 *Mesh of the ONERA M6 wing*

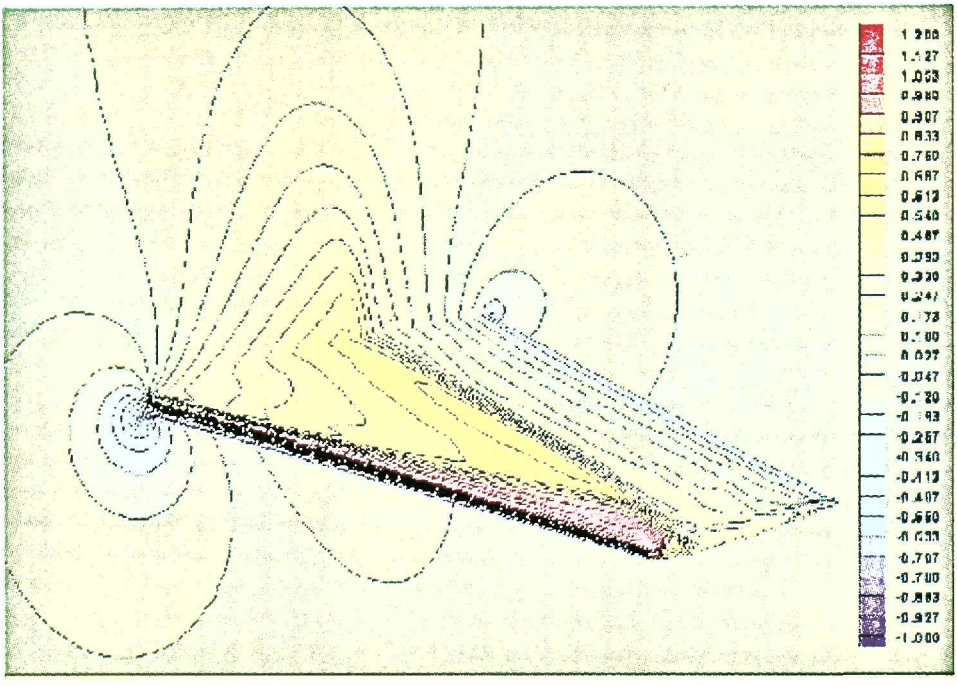


Figure 13.2 *Pressure distribution of the ONERA M6 wing in the symmetry plane and on the surface*

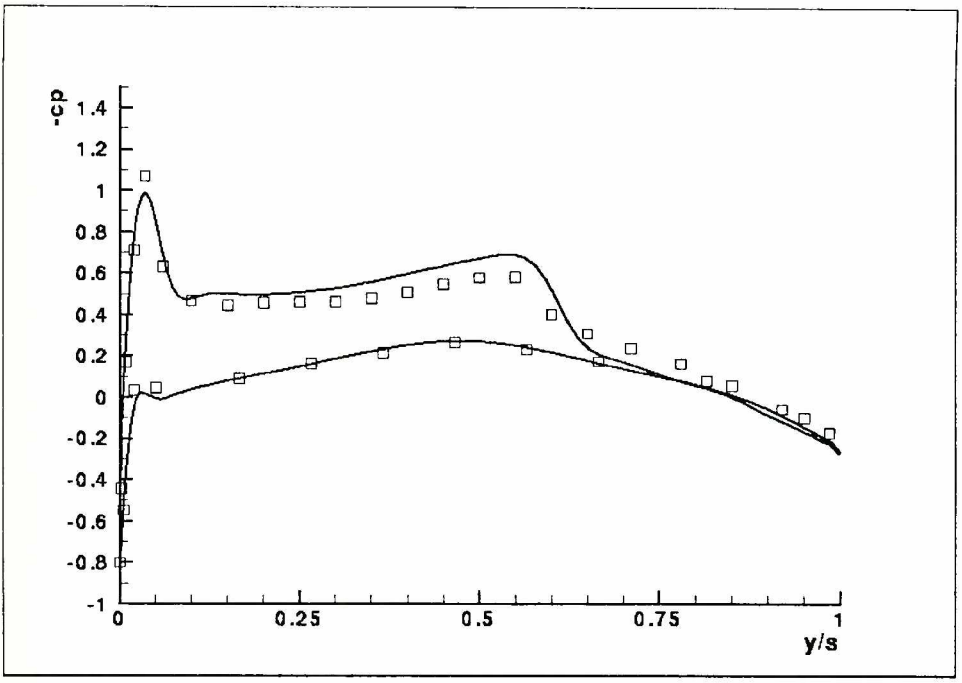


Figure 13.3 Pressure distribution of the ONERA M6 wing on the wing section y/s=0.2

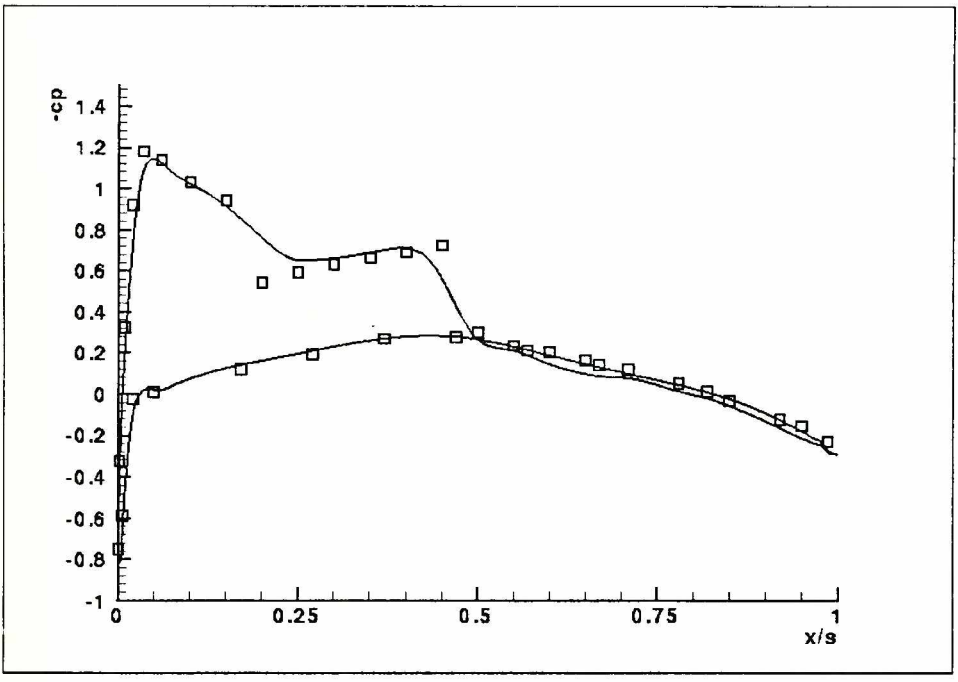


Figure 13.4 Pressure distribution of the ONERA M6 wing on the wing section y/s=0.6

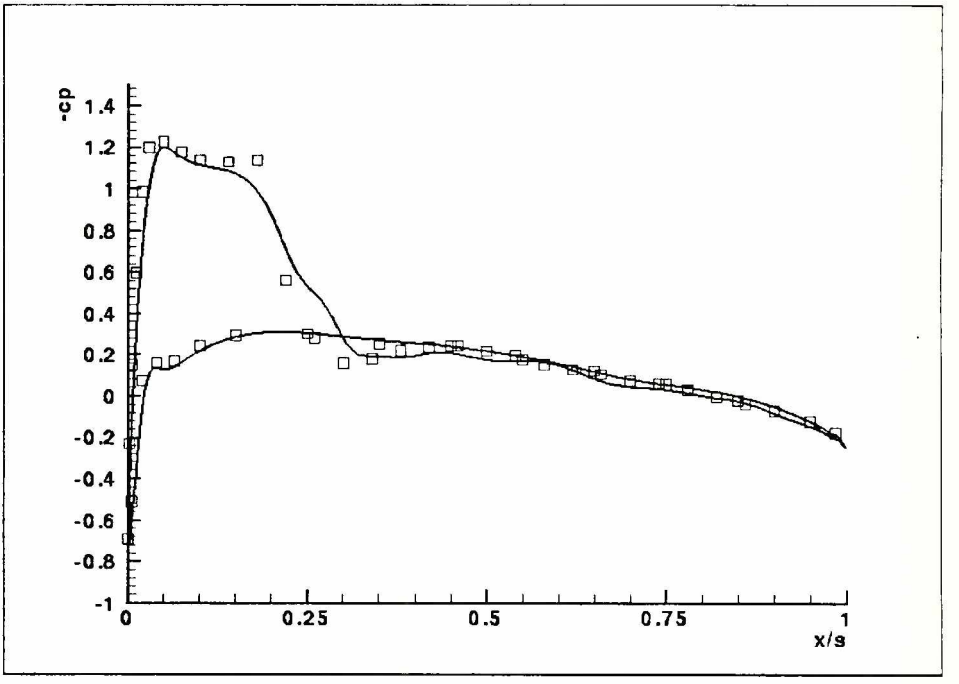


Figure 13.5 *Pressure distribution of the ONERA M6 wing on the wing section y/s=0.95*

13.2 Flow past the Space Shuttle

The hypersonic flow past the Space Shuttle has been investigated next. The mesh consists of about 200,000 points. The freestream Mach number was Ma = 10 and the angle of attack α = 30° while the freestream Temperature is T = 220K. The flow field has been calculated inviscid with the LUSSOR implicit solver and the USLIP high resolution scheme of Tatsumi et al. [25]. Two levels of multigrid have been used to speed up convergence. A total of 200 iterations were necessary to reduce the residuum by 6 order of magnitude. The surface mesh of the Space Shuttle is displayed in Figure 13.6. Please note that only half of the Space Shuttle has been calculated due to the symmetrical flow field calculation.

The density distribution on the lower part of the space craft as well as on the symmetry plane is shown in Figure 13.8. The bow shock formed in front of the vehicle can easily be observed. A better visualization of the shock structure is shown on Figure 13.7. The Mach number distribution at three stations along the fuselage indicates the spatial structure of the bow shock. In Figure 13.9 one can observe the pressure distribution on the lower part of the Space Shuttle.

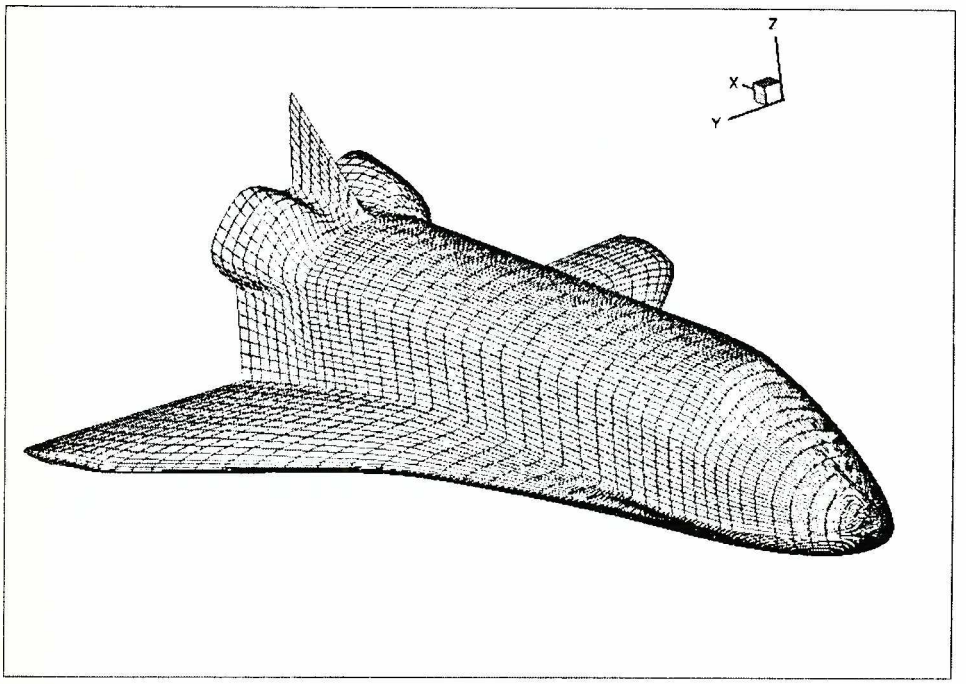


Figure 13.6 *Surface mesh on the Space Shuttle*

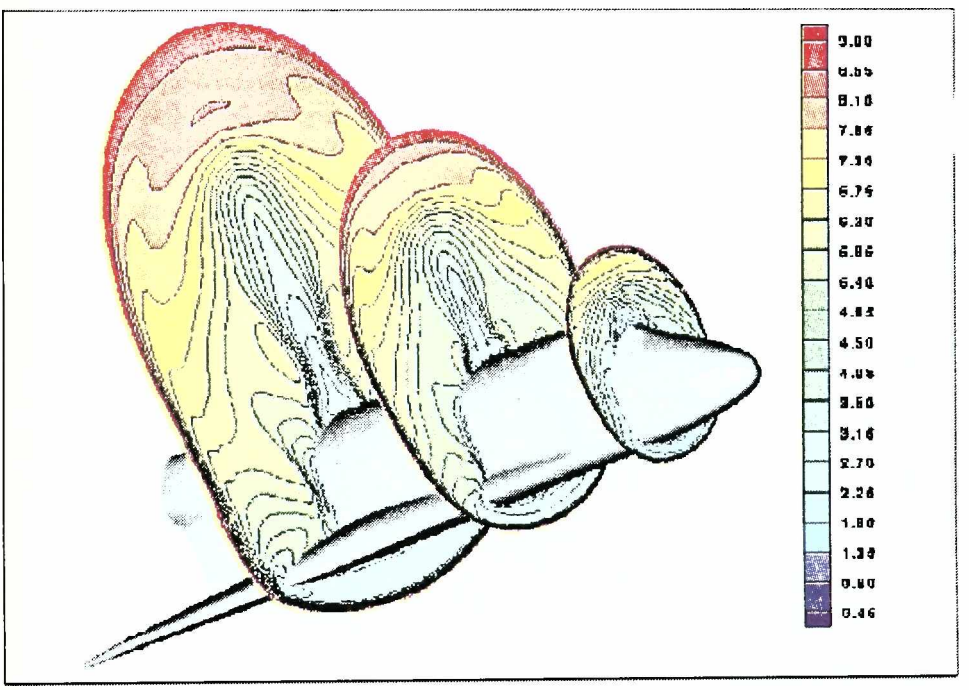


Figure 13.7 *Mach number distribution at three stations of the fuselage*

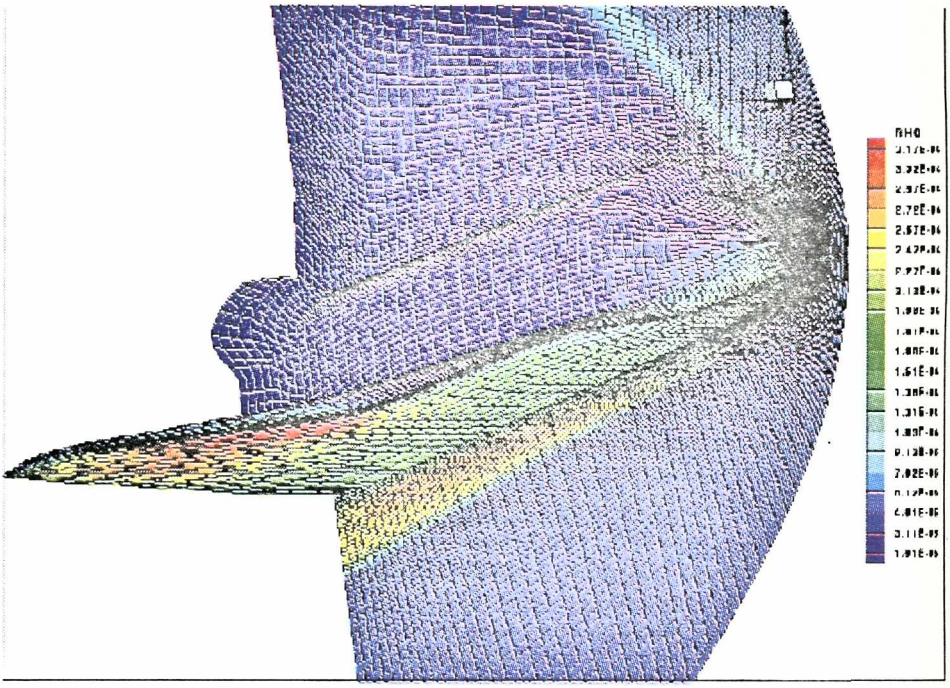


Figure 13.8 *Density distribution at the surface and in the symmetry plane of the Space Shuttle*

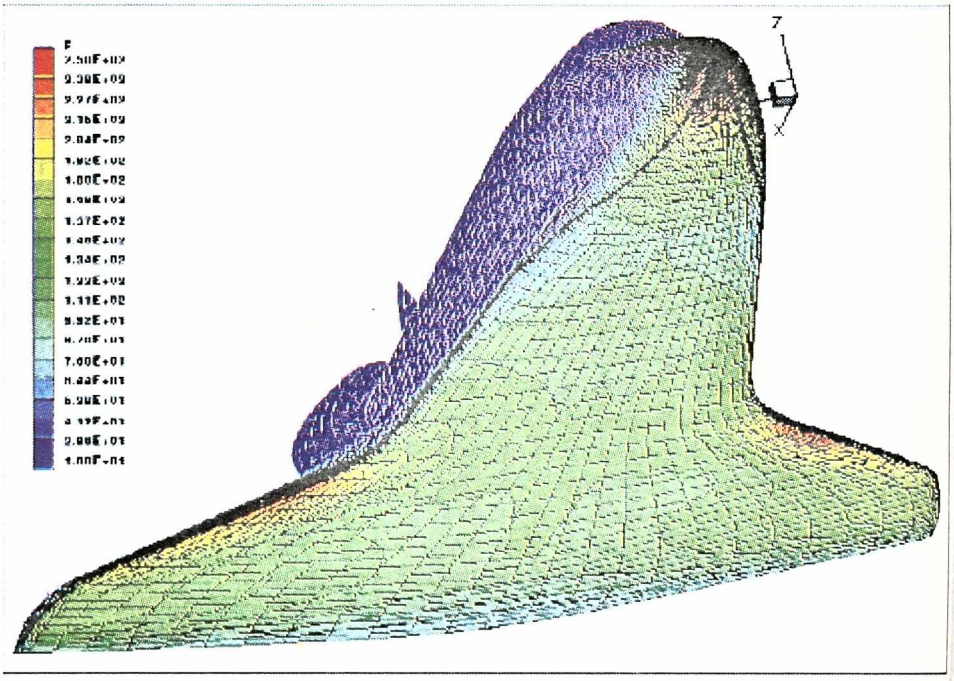


Figure 13.9 *Pressure distribution at the lower surface of the Space Shuttle*

13.3 Flow around a car

The flow around a basic car has been considered next. The mesh has been originally generated by Wüstenberg [15] at Volkswagen AG but smoothest by the author. All complex geometric details have been left out, the wheels are omitted and the wheelhouses closed, and the underbody of the model is flat. A region of 28m × 7m × 9.3m around the half model is discretized by 30 blocks with a total of about 2,500,000 cells in the finest mesh (see figure 3.11). The Reynolds number of $Re=7.62*10^6$ is based on the freestream velocity of u=32 m/s and the length of the model L=3.66m. The calculation has been done using the non-linear eddy viscosity model of Craft et al. with a freestream turbulence level of Tu=0.6 %. The pressure distribution in the symmetry plane and on the surface of the car as well as on the floor can by studied in figure 3.11 and in figure 3.12.

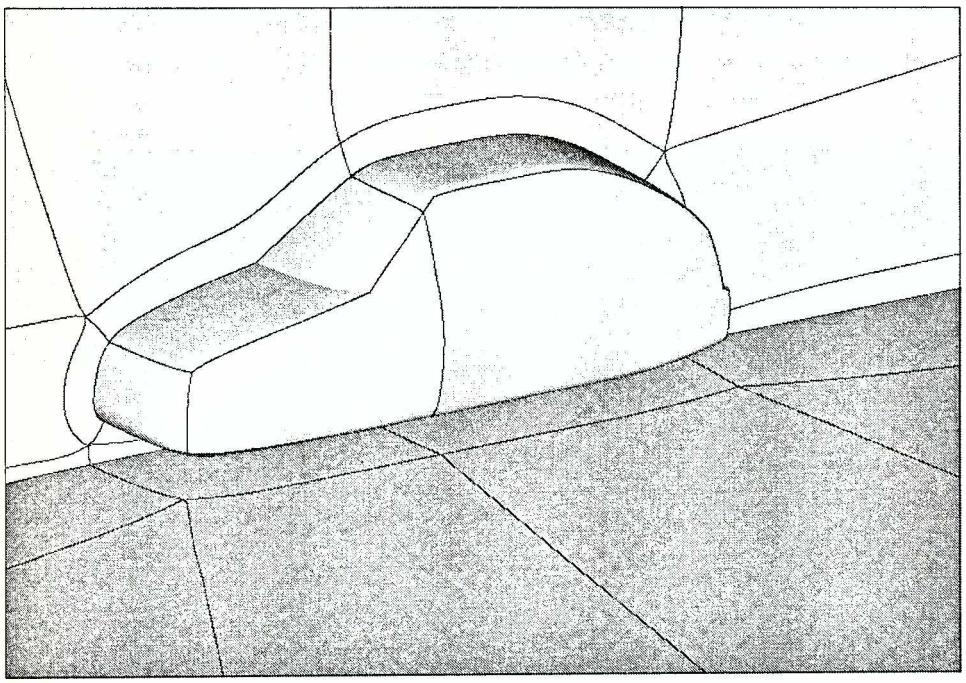


Figure 13.10 *Block structure of the VW-car*

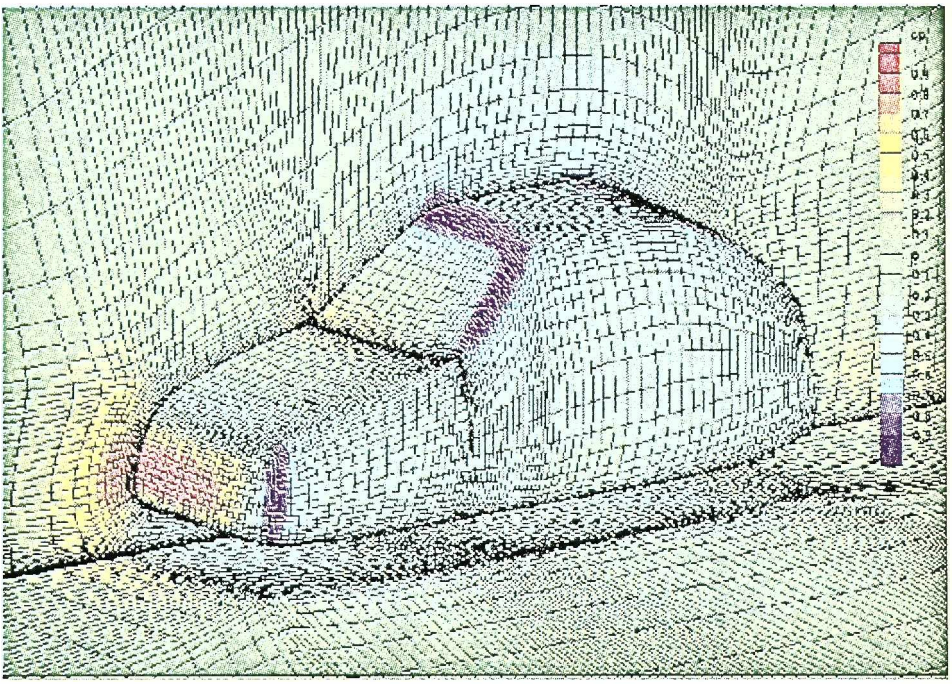


Figure 13.11 *Pressure distribution in the symmetry plane and on the surface of the VW-car*

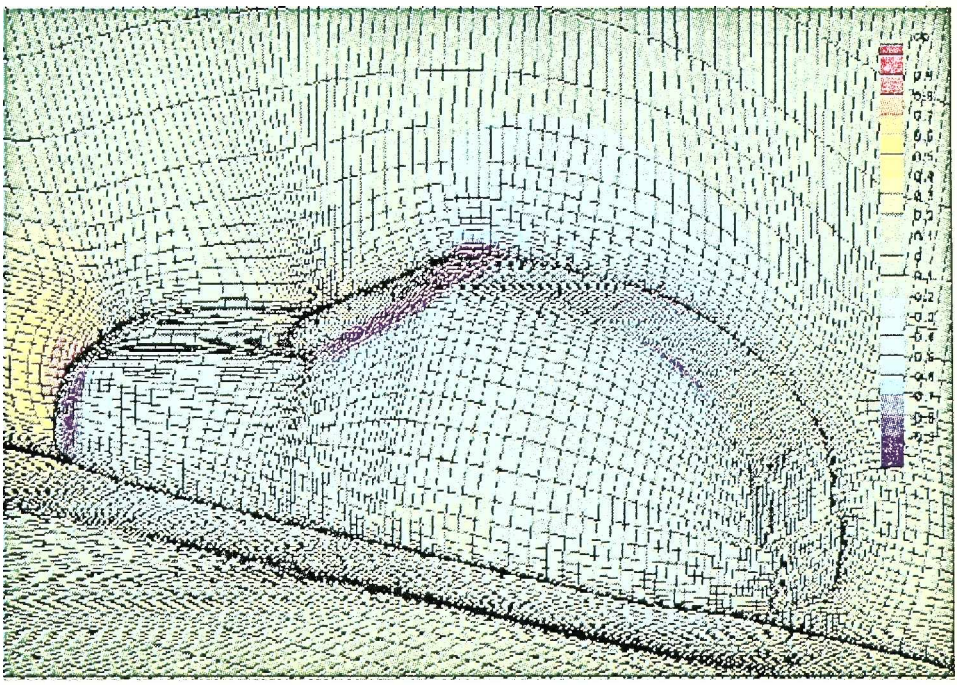


Figure 13.12 *Pressure distribution in the symmetry plane and on the surface of the VW-car*

13.4 Unsteady vortex sheeding past a cylinder

The unsteady flow past a circular cylinder measured by Cantwell and Coles [8] has been investigated next. The freestream Mach number is Ma=0.1 and the Reynolds number based on cylinder diameter is Re=140,000. An O-type grid has been generated with a total of about 50,000 points in the finest grid (see Figure 13.13). Calculations with appr. 12,000 points showed very nearly the same result as with 50,000 points indicating mesh independance.

Again the non-linear turbulence model of Craft et al. as well as the linear model of Launder/Sharma [41] have been used with a freestream turbulence level of Tu = 0.3 %. In order to advance the solution in time an accurate and efficient solver must be used. In KAPPA we have implemented the Dual Time Stepping scheme proposed by Jameson [6] and refined by Arnone [2]. Only 30 to 40 iterations per time step were necessary to reduce the residuum by six orders of magnitude. The velocity component in x-direction is shown in Figure 13.14 for the non-linear model. The vortex sheeding take place at a Strouhal number of St=0.215 for the calculation with the non-linear model and St=0.226 for the linear model while the experiment of Cantwell/Coles gave St=0.179.

The mean total drag calculated with the non-linear model was $c_d = 0.911$ while the linear model gave $c_d = 0.922$ compared to $c_d = 1.237$ in the experiment. The agreement with the experiment in this case is poor and depend very much on the transition criterion applied in the calculation. The pressure distribution around the circular cylinder is displayed on Figure 13.15. Both models show a lower pressure at $\phi = 90°$ and overpredict the pressure recovery in the wake.

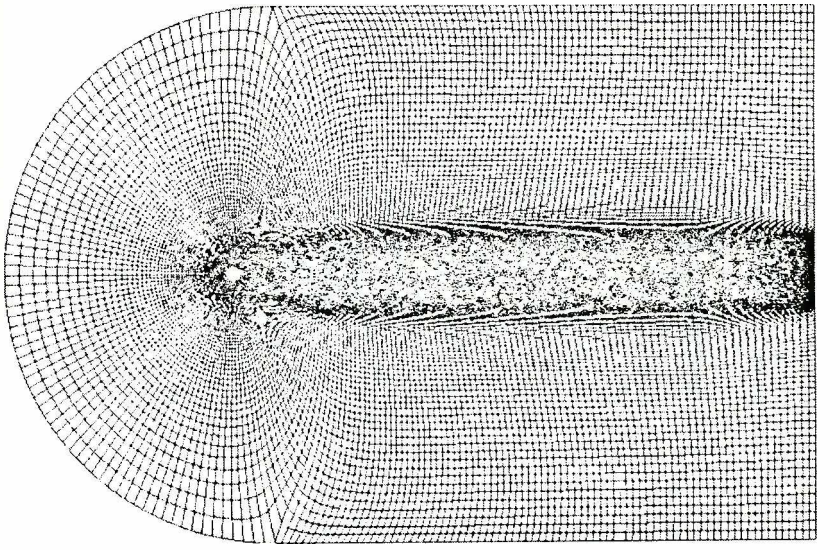


Figure 13.13 *Computational mesh around a circular cylinder*

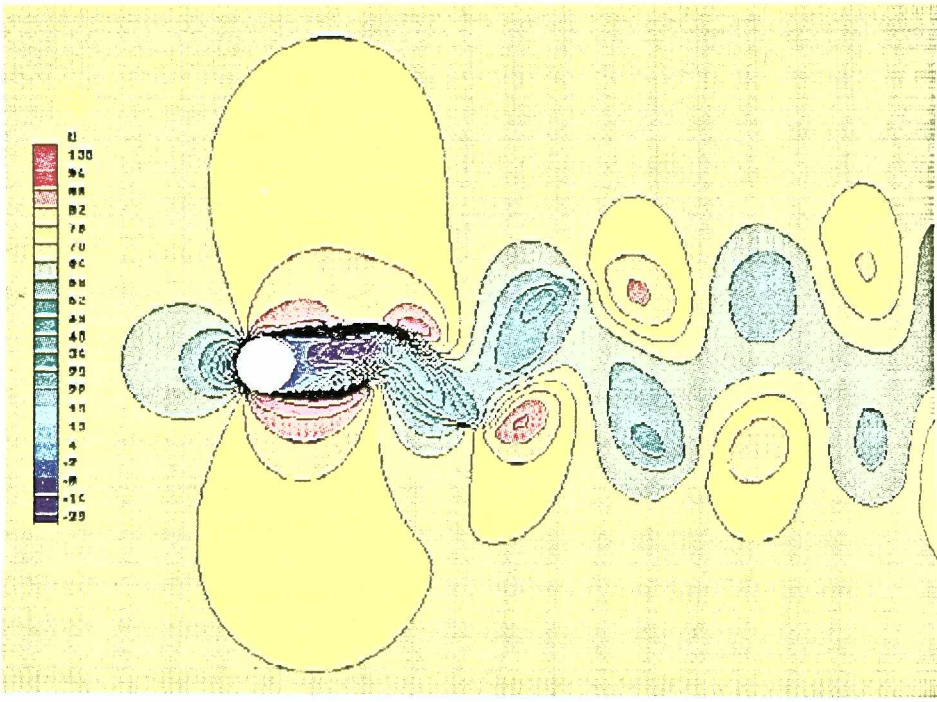


Figure 13.14 *Contours of const streamwise velocity-component*

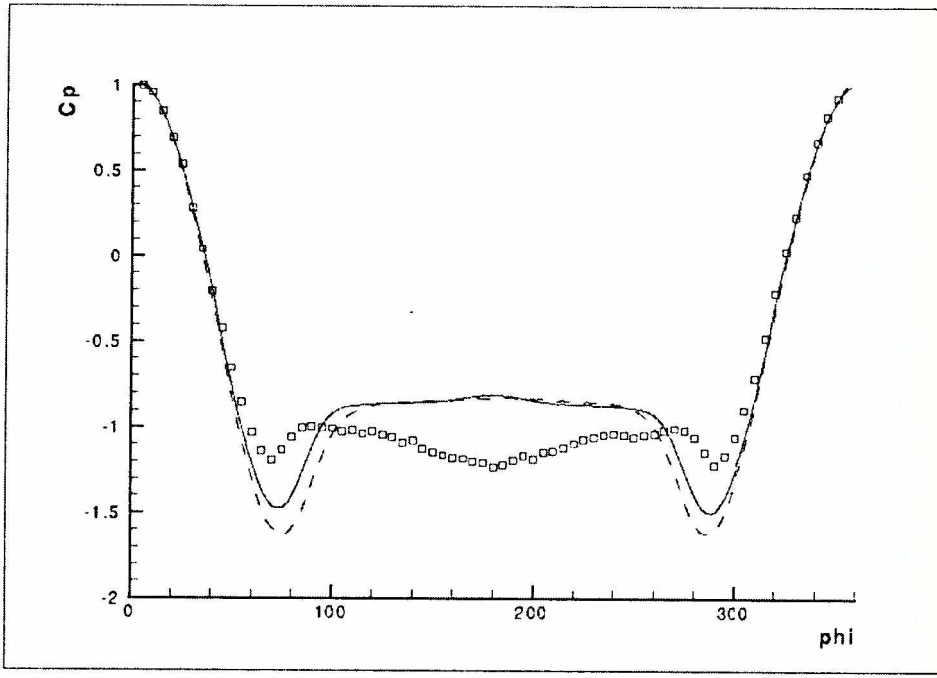


Figure 13.15 *Pressure coefficient distribution around cylinder*

13.5 Unsteady vortex sheeding past a VKI turbine blade

Finally the flow around a VKI turbine blade has been calculated. The measurements were conducted by Ubaldi et al. [7] of the University of Genova. The blade geometry is given by Cicatelli/Sieverding [10]. The Reynolds number based on the isentropic exit Mach number is Re = 1,600,000 and the exit Mach number Ma = 0.23. The turbulence level upstream of the blade has been measured as Tu = 0.8 %. The number of grid points for this test case is about 72,000 points on the finest grid. Calculations with about 18,000 points showed again almost the same results as in the finest grid.

The distribution of the Mach number is shown on Figure 13.18. One can recognize the vortex shedding at the round trailing edge of the turbine blade. The vortex shedding frequency is about 1500 Hz in the calculation while in the experiment [7] 1700 Hz has been measured. Another calculation with the linear eddy viscosity model of Launder/ Sharma gave no vortex shedding at all. The comparisons of the velocity profile at the suction side of the blade are shown next. The agreement of the calculated profile with the experiment is excellent at the station s/smax = 0.35 (Figure 13.16 and very good at the station s/smax = 0.95 in Figure 13.17.

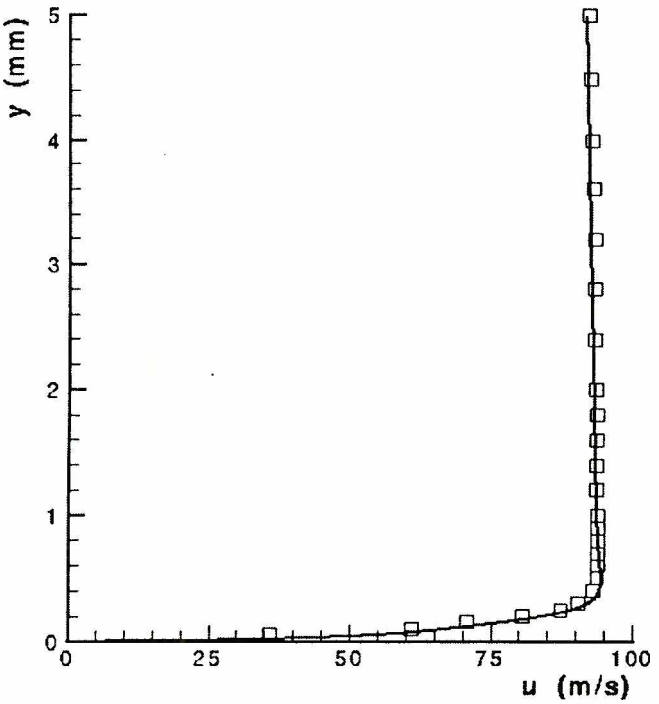


Figure 13.16 Suction side velocity profile at at s/smax = 0.35

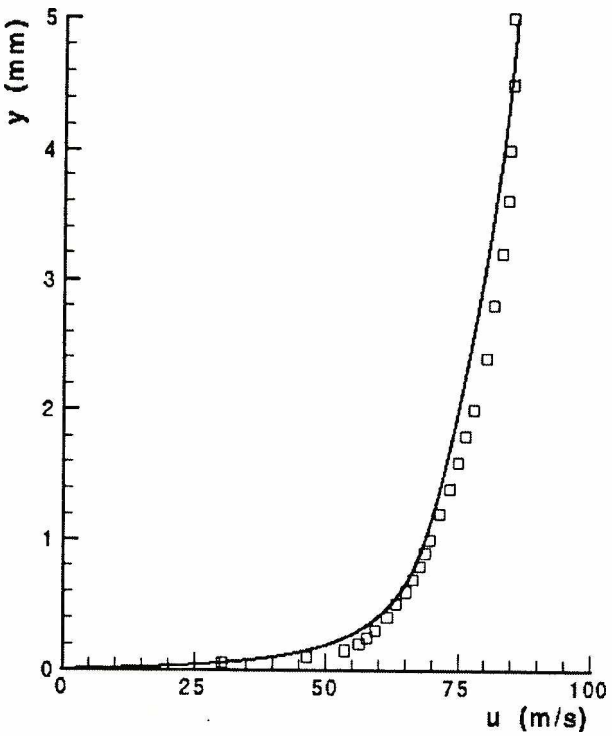


Figure 13.17 *Suction side velocity profile at at s/smax = 0.95*

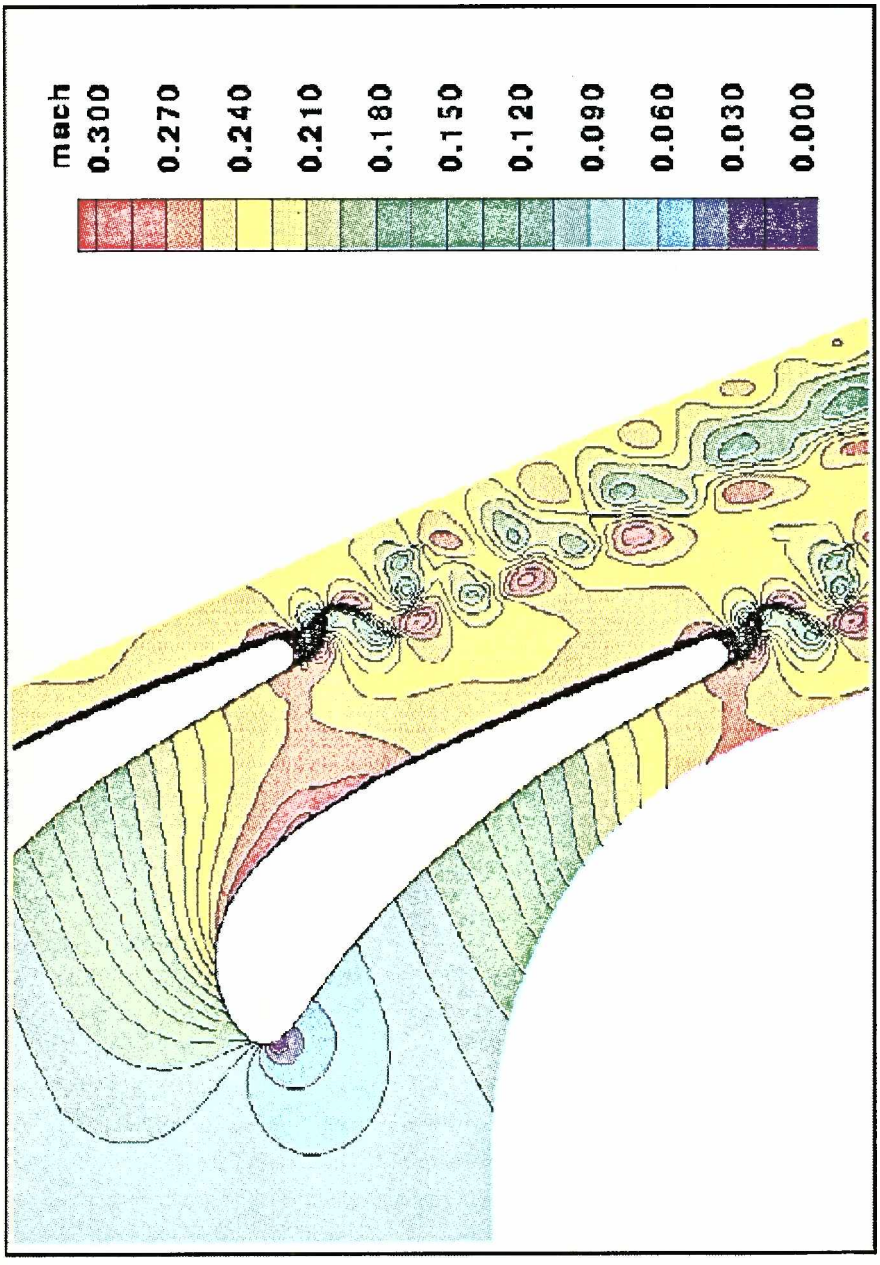


Figure 13.18 *Mach number distribution at the VKI turbine blade*

References

[1] AGARD138, *On Experimental Data Base for Computer Program Assessment*, AGARD-AR-138, Advisory Report 138, report of the FDP Working Group 04, 1979

[2] Arnone A., *Integration of Navier-Stokes Equations Using Dual Time Stepping and Multigrid Method*, AIAA-Journal, 1995

[3] Favre A., *Statistical equations of turbulent gases*, SIAM Problems of Hydrodynamics and Continuum Mechanics, 1969

[4] Jameson A., *Transonic Flow Calculation*, Technical Report MAE 1651, Princeton University, May 1984

[5] Jameson A., *Multigrid Algorithms for Compressible Flow Calculations*, Technical Report 1743, MAE-Report, 1985

[6] Jameson A., *Time Depentant Calculations using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings*, AIAA-paper, 1991

[7] Ubaldi M., Zunino P., Campora U., Ghiglione A., *Detailed Velocity and Turbulence Measurements of the Profile Boundary Layer in a Large Scale Turbunie Cascade*, ASME-International Gas Turbine and Aeroengines Congress and Exhibition, 1996

[8] Coles D., Cantwell B., *An experiment study of entrainment and transport in the turbulent near wake of a circular cylinder*, Journal of Fluid Mechanics, 1983

[9] Hirsch Ch., *Numerical computation of internal and external flows*, Vol I+II, John Wiley and Sons, 1990

[10] Sieverding K., Cicatelli G., *The effects of vortex shedding on the unsteady pressure distribution around the trailing edge of a turbine cascade*, ASME-paper no. 96-GT-359, 1996

[11] Suga K., Craft T. J., Launder B. E., *Extenting the Applicatility of Eddy Viscosity Models through the use of Deformation Invariants and Non-linear Elements*, 5th IAHR Conference on Refined-Flow Modelling and Turbulence Measurement, Paris 7 10 September, 1993

[12] Boussinesq D., *Theorie de L'ecoulement tourbillant*, Mem. Pres. Acad. Sci. XXIII, 1877

[13] Magagnato F., *Improvement of the efficiency and convergence of the Dornier flow solver IKARUS*, Technical Report BF 2/90B, Dornier Luftfahrt GmbH, 1990

[14] Magagnato F., *Validation of the K-ε model for 3d-flow fields*, Technical Report BF 2/90B, Dornier Luftfahrt GmbH, 1990

[15] Wüstenberg H., *Aerodynamic Computations on Basic Car Shapes*, 3rd International Conference on Innovation and Reliability in Automotive Design and Testing, 1992

[16] Thompson J. F., *Numerical Grid Generation*, J. Thompson Ed., 1982

[17] Lumley J. L., *Computational modelling of turbulent flows*, Advanced applied Mechanics, pages 123-176, 1978

[18] Hinze J. O., *Turbulence. Second Edition*, McGraw-Hill Publishing Company, 1975

[19] Ousterhout John K., *Tcl und Tk: Entwicklung graphischer Benutzerschnittstellen für das X Window System*, Addison-Wesley Publishing Company, 1995

[20] Martinelli L., *Calculations of viscous flow with a multigrid method*, Princeton University, Ph. D. Thesis, Mae Dept., 1987

[21] Eiseman P., Geometric method in computational fluid dynamics, *ICASE-Report 80-11*, 1980

[22] Radespiel R., A cell-vertex multigrid method for the Navier-Stokes equations, Technical Report TM-101557, NASA, 1989

[23] Fedorenko R. P., The speed of conver gence of one iterative process, *USSR Comp. Math. and Math. Physics*, 1964

[24] Leicher S., 3-D Navier-Stokes solution around the JRC Split-Sphere, Technical Report BF 8/88B, Dornier Luftfahrt GmbH, 1988

[25] Jameson A., Tatsumi S., Martinelli L., Flux-Limited Schemes for the Compressible Navier -Stokes Equations, *AIAA-Journal*, 1995

[26] Pulliam T. H., Euler and thin layer Navier-Stokes codes: ARC2D, ARC3D, *Computational Fluid Dynamics user's Workshop, Tennessee*, 1984

[27] Speziale C. G., Abid R., Anderson E. C., A critical evaluation of two-equation models for near wall turbulence, *ICASE Report No. 90-46*, 1990

[28] Stock H. W., Haase W., The determination of turbulent length scales in algebraic turbulence models for attached and slightly separated flows using Navier-Stokes methods, *AIAA-paper 89-1950, 1987*

[29] Haase W., Wagner B., Jameson A., Development of a Navier-Stokes Method based on a finite volume technique for the unsteady Euler equations, *5th GAMM-Conference on Numerical Methods in Fluid Dynamics, Rom*, 1983

[30] Martinelli L., Jameson A., Validation of a multigrid method for the Reynolds averaged equations, *AIAA-paper 88-0411*, 1988

[31] Rieger H., Jameson A., Solution of steady three-dimensional compressible Euler and Navier-Stokes equations by an implicit LU-scheme, *AIAA-paper 87-0619*, 1987

[32] Yoon S., Jameson A., An LUSSOR scheme for the Euler and Navier-Stokesequations, *AIAA-paper 87-0600*, 1987

[33] Launder B. E., Reynolds W. C., Rodi W., Mathiew J., Jeandel D., *Turbulence models and their applications*, Editions Eyrolles, Direction des Etudes et Recherche D'Electricite, 1985

[34] Baldwin B. S., Lomax H., *Thin Layer approximation and algebraic model for separated turbulent flows*, AIAA Paper 78-257, 1978

[35] Thompson J. F., Thames F. C., Mastin C. M., *Automatic numerical generation of body-fitted curvilinear coordinate systems for fields containing any number of arbitrary two-dimensional bodies*, Journal of Computational Physics, pages 299-319, 1974

[36] Yakhot V., Orszag S. A., *Renormalization group analysis of turbulence. I. Basic theory*, Journal of Scientific Computing, 1986

[37] Ferziger Joel H., Perić M., *Computational methods for fluid dynamics*, Springer Verlag, 1996

[38] Anderson D. A., Tannehill J. C., Pletcher R. H., *Computational Fluid Mechanicsand Heat Transfer*, McCraw-Hill, 1984

[39] Gibson M. M., Rodi W., *A Reynolds stress closure model of turbulence applied to the calculation of highly curved mixing layers*, Journal of Fluid Mechanics, pages 161-182, 1981

[40] Launder B. E., Reece G. J., Rodi W., *Progress in the development of a Reynoldsstress turbulence closure*, Journal of Fluid Mechanics, pages 537-586, 1975

[41] Launder B. E., Sharma B. I., *Application of the energy dissipation model of turbulence to the calibration of flow near a spinning disc*, Letter Heat Mass Transf pages 131-138, 1976

[42] Cebeci T., Smith A. M. O., *Analysis of turbulent boary layers*, Academic Press., 197

[43] Launder B. E., Spalding D. B., *Mathematical models of turbulence*, Academic Press London and New York, 1972

[44] Patel V. C., Rodi W., Spalding D. B., *Turbulence models for near-wall and low Reynolds number flows: A Review*, AIAA-Journal, pages 1308-1319, 1985

[45] Jameson A., Turkel E., I*mplicit schemes and LU-decomposition*, Mathematical Computations, 1981

[46] Jameson A., Schmidt W., Turkel E., *Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes*, AIAA-paper 81-1259, 1981

[47] Swanson R. C., Turkel E., *Artificial dissipation and central difference schemes for t Euler and NS-equations*, AIAA-paper 87-1107, 1987

[48] Martinelli L., Yakhot V., *RNG-Based turbulence transport approximations with applications to transonic flows*, AIAA Paper 89-1950, 1989

[49] Jameson A. Yoon S., L*U-implicit schemes with multiple grids for the Euler equatio* AIAA-paper 86-0105, 1986

[50] Rodi W., *A new algebraic relarion for calculating the Reynold stresses*, ZAMM 56 T21g-T221, 1976

[51] Rodi W., *Examples of turbulence models for incompressible flows*, AIAA-Journal, pages 872-879, 1982

[52] Seibert W., *A graphic interactive program-system to generate block structured volume grids for fluid flow analysis, Functional description, DOGRID Version 5*, D Report, No. BF 8/90B, 1982