Deutsche Foschungsanstalt
für Luft- und Raumfahrt

DLR

Forschungsbericht

A Multiple-Block Multigrid Method for the Solution of the Three-Dimensional Euler and Navier-Stokes Equations

Harold Atkins

DLR
Institut für Entwurfsaerodynamik
Braunschweig

DLR-FB 90-45

(Als Manuskript gedruckt)

Herausgegeben von
der Deutschen Forschungsanstalt für Luft- und Raumfahrt e. V. (DLR).
Mitglied der Arbeitsgemeinschaft der Großforschungseinrichtungen (AGF).

Zu beziehen durch
Wissenschaftliches Berichtswesen der DLR
Postfach 90 60 58, 5000 Köln 90.
ISSN 0171-1342

Deutsche Forschungsanstalt
für Luft- und Raumfahrt

DLR

Forschungsbericht

A Multiple-Block Multigrid Method for the Solution of the Three-Dimensional Euler and Navier-Stokes Equations

Harold Atkins

DLR
Institut für Entwurfsaerodynamik
Braunschweig

31 pages
14 figures
8 references

DLR-FB 90-45

Dm 12,50

Manuskript eingereicht am 9. Oktober 1990

A Multiple-Block Multigrid Method for the Solution of the Three-Dimensional Euler and Navier-Stokes Equations

DEUTSCHE FORSCHUNGSANSTALT FÜR LUFT- UND RAUMFAHRT

Forschungsbereich Strömungsmechanik
Institut für Entwurfsaerodynamik
Flughafen, D-3300 Braunschweig, FRG

Braunschweig, im September 1990

Institutsdirektor:
Dr.-Ing. H. KÖRNER

Verfasser:
Dr. H. ATKINS *

)*
Dr. H. ATKINS, Gastwissenschaftler am Institut für Entwurfsaerodynamik in einem Austauschprogramm zwischen DLR und NASA, gegenwärtige Adresse: NASA Langley Research Center, Hampton, VA 23665-5225

Multiple block, multigrid, Runge-Kutta scheme, Euler equations, Navier-Stokes equations

A Multiple-Block Multigrid Method for the Solution of the Three-Dimensional Euler and Navier-Stokes Equations

Summary

A multiple-block multigrid method for the solution of the 3-D Euler and Navier-Stokes equations is presented. The basic flow solver is a cell-vertex method which employs central difference spatial approximations and Runge-Kutta time stepping. The use of local time stepping, implicit residual smoothing, multigrid techniques and variable-coefficent numerical dissipation results in an efficient and robust scheme. The multi-block strategy places the block loop within the Runge-Kutta loop such that accuracy and convergence are not effected by block boundaries. This has been verified by comparing the results of one- and two-block calculations in which the two-block grid is generated by splitting the one-block grid. Results are presented for both Euler and Navier-Stokes computations of wings and wing/fuselage combinations.

Blockstruktur, Mehrgitter, Runge-Kutta Schema, Euler-Gleichungen, Navier-Stokes-Gleichungen

Ein blockstrukturiertes Mehrgitterverfahren für die Lösung der dreidimensionalen Euler- und Navier-Stokes-Gleichungen

Übersicht

In der vorliegenden Arbeit wird ein blockstrukturiertes Mehrgitterverfahren für die Lösung der 3-D Euler und Navier-Stokes-Gleichungen vorgestellt. Der Strömungslöser besteht aus einer Zelleckpunktdiskretisierung mit zentralen Differenzen und einem Runge-Kutta-Zeitschrittverfahren. Mit Hilfe von lokalen Zeitschritten, einer impliziten Glättung der Residuen, eines Mehrgitteralgorithmus' und künstlichen dissipativen Termen mit variabler Skalierung erhält man ein effizientes und robustes Verfahren. Bei Verwendung mehrerer Rechenblöcke wird die Rechenschleife über die Blöcke innerhalb der Schleife über die Stufen des Runge-Kutta-Schemas angeordnet, so daß Genauigkeit und Konvergenz des Verfahrens nicht durch die Blockgrenzen beeinträchtigt werden. Dieses kann anhand von Rechnungen mit einem bzw. zwei Rechenblöcken gezeigt werden, wobei das Netz mit zwei Blöcken durch Aufteilung des Netzes mit einem Block erzeugt wurde. Es werden Ergebnisse für Flügel und Flügelrumpfkombinationen und Lösungen der Euler- und Navier-Stokes-Gleichungen angegeben.

Contents

Introduction

Advances in computers and algorithms have reduced the cost of simulating complex flows to within reasonable values. However most calculations of practical interest still pose problems either due to their shear size, or their geometric complexity, and sometimes both. A technique that greatly eases this restriction is the use of multi-block strategy. In this approach, the physical domain is subdivided into several smaller parts which have simpler topologies and are accurately modeled by a manageable number of points. The grid generation also becomes easier because of the simplier topologies in each block. Relaxing constraints on the connectivity between block can further simplify the grid generation stage. In most cases, however, techniques that simplify the generation of the grid usually complicate the implementation of the flow solution algorithm.

The primary objective of this work is to implement a proven flow solver in a multi-block frame work while preserving as nearly as possible its accuracy and convergence properties. The basic flow solver[1] has been validated for three-dimensional flows over wings and found to be accurate and efficient for single-block domains. The solver is a cell-vertex method which uses central spatial differences and Runge-Kutta time stepping to solve the Euler or thin-layer Navier-Stokes equations. Several acceleration techniques are applied to improve the efficiency. Among these are: multigrid, local time-stepping, enthalpy damping (for inviscid cases), implicit residual smoothing, and blended second and fourth difference numerical smoothing. Turbulent flow calculations employ a Baldwin-Lomax model.

The multi-block algorithm presented here completely preserves the accuracy and convergence properties of the base solver. Block loops which are placed within the Runge-Kutta loop, combined with strict treatment of cut ghost point data, results in a robust algorithm. The algorithm is implemented to support in-core solutions, or out-of-core solutions using a SSD or similar device, with an acceptable overhead.

A secondary objective is to make the implementation as topology independent as possible without making the program too complicated to use. This is achieved by developing a flexible data structure to describe and control all boundary conditions and block mappings.

The method is validated by comparing results from one and two block domains to the same problem. Single block grids are split in different ways so that the global domain is identical. Solution are presented for viscous and inviscid wings, and for an inviscid wing-fuselage combination. Comparisons are made with respect to the accuracy and efficiency.

A brief description of the governing equations and the basic flow solver is given; however, more detail may be found in reference 1. Emphasis is placed on the multi-block aspect of the algorithm, which is discussed in detail.

Governing Equations

The normalized integral form of the mass-averaged Navier-Stokes equations can be written as

$$\frac{\partial}{\partial t}\iiint_{\Omega} U\,dv = -\iint_{\partial\Omega} \vec{F}\cdot\vec{n}\,ds \qquad 1$$

where

$$U = (\rho, \rho u, \rho v, \rho w, \rho E)^T$$

The variables ρ, u, v, w, and E denote the density, the Cartesian velocity components, and the specific total internal energy. When appropriate, the Cartesian velocity components are also denoted as u_1, u_2, u_3. The control volume is denoted by Ω, its boundary surface by $\partial\Omega$, and the unit outward normal by $\vec{n}$. The flux dyadic $\vec{\vec{F}}$ is divided into its Euler (e) and viscous (v) components.

$$\vec{\vec{F}} = \vec{\vec{F}}_e - \vec{\vec{F}}_v$$

$$\vec{\vec{F}}_e = \begin{bmatrix} \rho\vec{V} \\ \rho u\vec{V} + p\vec{i} \\ \rho v\vec{V} + p\vec{j} \\ \rho w\vec{V} + p\vec{k} \\ \rho H\vec{V} \end{bmatrix} \quad \text{and} \quad \vec{\vec{F}}_v = \begin{bmatrix} 0 \\ \vec{i}\cdot\vec{\vec{\tau}} \\ \vec{j}\cdot\vec{\vec{\tau}} \\ \vec{k}\cdot\vec{\vec{\tau}} \\ \vec{V}\cdot\vec{\vec{\tau}} + \nabla\cdot T \end{bmatrix}$$

where $\vec{V}$ is the Cartesian velocity vector and $\vec{i}$, $\vec{j}$, and $\vec{k}$ are the Cartesian unit vectors (also denoted as $\vec{i}_1$, $\vec{i}_2$, $\vec{i}_3$). The shear stress dyadic is defined in terms of tensor notation as

$$\vec{\vec{\tau}} = \left[\vec{i}_i \tau_{i,j} \vec{i}_j\right] \qquad \tau_{i,j} = \mu\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}\right]$$

The temperature T pressure p and enthalpy are related to the dependent variables through the following algebraic relations:

$$T = (\gamma-1)\left[E - \frac{\vec{V}\cdot\vec{V}}{2}\right]$$

$$H = E + T \qquad \text{and} \qquad p = \rho T$$

The thermodynamic variables, P, T, and ρ are normalized by their freestream states P_∞, T_∞, and ρ_∞. The velocity components are normalized by $\sqrt{\gamma P_\infty/\rho_\infty}$. Employing the empirical power rule for viscosity results in the following relations for the normalized viscosity and conductivity:

$$\mu = \frac{\gamma^{1/2}M_\infty}{Re_\infty}(T/T_\infty)^{0.75} \qquad \text{and} \qquad k = \frac{\gamma}{\gamma-1}\frac{\mu}{Pr}$$

Turbulence is modeled through the introduction of a turbulent viscosity μ_t. In the shear stress terms, the laminar viscosity is replaced by $\mu+\mu_t$, and the conductivity is modified by replacing μ/Pr with $\mu/Pr+\mu_t/Pr$.

The Solution Algorithm

The computation domain is formed by partitioning the physical domain into hexahedrons to form a structured grid or several connecting structured grids. Discrete point values of the solution vector are stored at each vertex of the grid. Solutions to equation 1 are obtained by approximating that equation at each vertex and integrating in time to obtain a steady state. For the sake of efficiency, the correct time evolution is altered by the application of several acceleration techniques. The solution algorithm follows the method of lines[2] in which the time derivative is integrated as an ODE subject to the spatial terms as a forcing function.

$$\frac{d}{dt}\iiint_{\Omega} \mathbf{U}\,dv = -\mathbf{R}(\mathbf{U}) \qquad 2$$

and

$$\mathbf{R}(\mathbf{U}) = \iint_{\partial\Omega} \vec{\mathbf{F}}\cdot\vec{n}\,ds \qquad 3$$

For convenience of discussion, the spatial terms, or residuals, are further divided into contributions from the Euler, viscous and numerical smoothing terms.

$$\mathbf{R}(\mathbf{U}) = \mathbf{R}_e(\mathbf{U}) + \mathbf{R}_v(\mathbf{U}) + \mathbf{R}_s(\mathbf{U})$$

Equation 2 is integrated in time using a five-stage Runge-Kutta method.

$$\mathbf{W}^0 = \mathbf{U}^n$$

$$\mathbf{W}^k = \mathbf{W}^0 - \alpha_k \nabla t\, \mathbf{R}^{k-1} \qquad k=1,2,3,4,5$$

$$\mathbf{U}^{n+1} = \tilde{L}\cdot\mathbf{W}^5$$

where

$$\mathbf{R}^k = L\cdot\hat{\mathbf{R}}^k \qquad \hat{\mathbf{R}}^k = \mathbf{R}_e(\mathbf{W}^k) \quad + \mathbf{R}_v(\mathbf{W}^0) + \sum_{m=0}^{k} \gamma_{k,m}\mathbf{R}_s(\mathbf{W}^m)$$

The viscous residual is computed only during the first stage and held fixed thereafter. The numerical smoothing terms of a given stage are a linear combination of the smoothing over several stages[3]. Values for the parameter α are

$$\alpha_1 = 1/4,\ \ \alpha_2 = 1/6,\ \ \alpha_3 = 3/8,\ \ \alpha_4 = 1/2,\ \ \alpha_5 = 1.$$

The operators L and $\tilde{L}$ denote the accumulated effect of boundary conditions and acceleration techniques.

The spatial terms are constructed centrally about each point in the computational domain. Each point lies at the intersection of eight cells which when combined form a super-cell, Fig. 1. The small cells will be referred to as the compact cells. The super-cell outlines the extent of the stencil of the physical spatial terms.

The inviscid residual of the super-cell is formed by first computing the residual of each compact cell, and then summing over the eight contributing cells. The flux through any face of a compact cell is computed from the arithmetic average of the conserved variables at the corners.

The contribution of the viscous term is computed using an auxiliary cell about the point. The vertices of this cell are the centers of the eight compact cells comprising the super-cell. The face normal vectors for the auxiliary cell are computed by averaging the normal vectors of the nearby compact cell faces. The gradients on the face of the auxiliary cell are computed using a local coordinate transformation.

$$\frac{\partial}{\partial \gamma} = \frac{\partial \xi}{\partial \gamma}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial \gamma}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial \gamma}\frac{\partial}{\partial \zeta}$$

for γ = x, y, or z. Only the gradients which are normal to the face in the cell and are in a selected direction are accounted for, resulting in the thin-layer approximation. Multiple directions may be selected. Derivatives at the face are computed by second order central differences. Scalar quantities are computed from second order averages. The viscosity is computed from the average temperature.

The numerical smoothing employed is based on the formulation of Jameson, Schmidt, and Turkel[4]. It is a combination of second and fourth derivative operators with coefficients which depend on the pressure gradient, the acoustic wave speeds, and the cell aspect ratios. The operator is formulated as a conservative one-dimensional operator for each coordinate direction.

Boundary Conditions

The present method supports five physically different boundary conditions: slip and no-slip walls, far-field conditions, symmetry planes, and cuts or mapped boundaries. The boundary conditions are implemented in a flexible data structure that imposes no assumptions on the topology. The implementation combines a variety of strategies, including the use of ghost cells, and impacts the flow solver at several stages.

For the present, it is useful to consider the topological arrangement of boundaries of cell vertex methods and compare them with the more common cell centered control volume approach. Figure 2 illustrates both approaches for two common situations; the leading and trailing edge regions of a wing with an H-type mesh. The grids in both cases are identical; however the orientation of the flow variables is crucially different. The flow variables for the cell vertex method coincide with the grid points, whereas for the cell centered method the flow variables are located at the center of the cell (or are an average over the cell). As a

consequence, the cell centered method has a natural alignment between the flow variable and the control volume, and between the boundaries of the control volume and the domain boundaries. In contrast, flow variables of the cell vertex method can lie on the interface of two different boundary types, and the super-cell can extend outside the domain as well as overlap boundary types.

The differences do not cause difficulties but they must be taken into consideration when applying the boundary conditions. In addition to setting the flow variables at ghost points, the implementation of boundary conditions involves modifications to the inviscid compact residuals, the numerical smoothing, and super-cell residuals. In the following sections each type of boundary condition is briefly discussed.

Slip wall conditions require no flow through the surface. For the compact inviscid residual of ghost cells lying on a boundary of this type, the component of momentum normal to the boundary is reflected from the neighboring interior cell. Additionally the super-cell residual is modified by a similar projection such that the normal component of the solution is zero after the each Runge-Kutta stage. The ghost point values primarily influence the numerical smoothing terms. Thus first order extrapolation is adequate to set the flow variables.

The no-slip condition requires the velocity to be zero at the boundary. All momentum components of the compact inviscid residual of ghost cells on a no-slip wall are reflected, and all similar components of the related super-cell residuals are zeroed. The flow variables at ghost points are set by reflecting the velocities and thermodynamic quantities are treated as if symmetric.

The symmetry condition requires the flow to be symmetric with respect to a specified plane. A direct consequence is that there is no flow through the plane, similar to a slip wall. The operations to the compact and super cell residuals are identical to those of slip walls except the surface normal vector is constant over the surface. Flow variables at ghost points are reflected from the interior points. Polar singularities are usually treated as symmetry planes.

Far field boundary conditions communicate the influence of the freestream Mach number, enthalpy, and flow angle to the computational domain. For subsonic inflow/outflow boundaries, a locally one-dimensional characteristic approach[5] is used to set the value of flow variables at boundary and ghost points. The super-cell residuals at far field boundary points are zeroed and the flow solver discussed previously is not applied. The equations are linearized about the values at the interior point nearest to the boundary, and characteristic quantities are extrapolated from either the interior or the far field depending on the direction of the flow through the boundary. In inviscid calculations, the energy is computed such that freestream enthalpy is enforced. This is essential if enthalpy damping is to be used. The induced velocities due to lift are be accounted for through the use of a compressible lifting line theory.

Cut boundaries communicate data between two computational blocks or between disconnected regions in the same computational block. Ghost points associated with cut boundaries correspond to a point somewhere within the interior of the physical domain. In the present implementation, the mapping is assumed to be "one to one and onto". The condition is implemented by copying values from the interior point to the ghost point. In multi-block out-of-

core calculations the interior point is not readily available. The data required to update a cut is stored on a record addressable file and is updated after each Runge-Kutta stage as well as after the multigrid injection and prolongation operations. The sequence for updating ghost points is critical to the preservation of the convergence properties of the basic flow solver. This is discussed in more detail in the following section.

Multi-Block Algorithm

In a multi-block approach, the global domain is divided into several smaller blocks for which grids are more easily generated. The solution algorithm is applied to each block in some prescribed sequence. A variety of multi-block strategies have been applied to computational methods. The main issue is at what depth within the algorithm should the block loop be placed, and how often should data be transferred between blocks. The simplest approach to implement is to wrap a block loop around the complete flow solver. If applied to the present flow solver, for example, one complete multigrid cycle would be performed in a given block before moving on to the next. In this approach, it is impossible to maintain constant communication of data between the block interfaces. Consequently, ghost point data at the cuts must often lag, or sometimes lead, the data at the interior points. This approach works well with equations which can be solved by relaxation methods, but is not suited for fast time-asymptotic methods. A single-grid version of the present flow solver was used in a multi-block algorithm of this type[6,7]. Accurate solutions were obtained; however, the number of time steps that could be performed in each block varied with the severity of the case. In another instance involving a time-asymptotic multigrid algorithm[8], it was necessary to conclude the calculation with a significant number of single-grid time steps in order to converge the solution. As the convergence rate of the basic flow solver increases, the importance of block communication also increases.

In the present approach, the objective is to preserve the accuracy and rapid convergence properties of the basic flow solver. To do this the block loop is placed deep within the algorithm. Although this introduces some computational overhead, the resulting multi-block algorithm closely simulates the results of a single block calculation. The method has been implemented to either run in-core on a large memory machine or out-of-core using a solid-state-disk (SSD) or similar device. The following description of the method is broken into three topics: 1) the block solution algorithm, 2) the data structure of field variables, 3) the data structure and control of boundary conditions.

The Block Solution Algorithm

The program structure is on three levels. At the top level are routines that control the multigrid strategy, and thus, the grid level. At the middle level are routines controlling the block loops, and at the bottom are block structured routines which perform an operation on a given block and level. Figure 3 shows an approximate block diagram of the multi-block solution algorithm. The left side of the figure illustrates the major steps of the multigrid algorithm while one Runge-Kutta time step is expanded on the right side. The operations indicated within each box are performed for each block of data before proceeding to the next block. Each stage of the Runge-Kutta time step is in a separate block loop with the boundary

conditions applied in two passes, before and after each stage. This is done to ensure that all variables are at a consistent level at the start of each stage of the time step. More explained further in the section on boundary conditions.

Data Structure

The data structure accommodates in-core computations or, for large problems or small machines, out-of-core computations. For in-core computations, all levels and all blocks of each field variable are stored in a long array. A separate pointer array stores the starting point of the data for each block and level. All data required for a block operation is passed into the low-level block structured routine through the argument list. In principle the low-level routine does not know or need to know on which block it is operating.

For out-of-core computations, only the field variables for a single block are stored in core at any given time. As before, for each block there is a pointer array which stores the starting point of each level. The data for all blocks and levels are stored on two sets of files. The use of two sets of files, as shown in fig. 4, eliminates the need for using slower record addressable I/O (input/output), and permits the I/O to be performed synchronously. The required data is read from the 'INPUT' file(s), a block operation is performed, and the resulting data is written to the 'OUTPUT' file(s). At the end of the block loop, all files are rewound and files to which new data was written are switched with the corresponding 'INPUT' file. The amount of I/O is limited by reading only the data needed to perform the block computation, and writing only the results of the computation. This is done by distributing the field variables across a set of files, instead of a single large file. For instance, when computing the multigrid corrections, only the first and last solutions on a given level are needed; residuals, smoothing coefficients, cell normals etc. are not needed and should not be read or written. I/O is also limited by allocating a different set of files for each level. Some data quantities, such as geometric data, are required only as input to a calculation and are never altered. As such, only an 'INPUT' file is needed. At present, the number of files required for the field variables is 5 times the number of multigrid levels.

Boundary Conditions

The correct sequencing of the boundary conditions is perhaps the most important aspect of the multi-block algorithm. The basic flow solver is a time-asymptotic method. Its stability and convergence properties are strongly tied to the assumption that all variables entering into the calculation of a spatial derivative are at the same stage and time level. When a ghost point is allowed to lag or lead the interior data, the derivative computed near the boundary looses its physical significance. Since the deviation caused by the lag or lead is tied directly to the rate at which the flow is changing, a rapidly converging solution will be most seriously affected. In the present algorithm, the lag and lead affects are completely eliminated through the following three steps.

The first step is to ensure that all the boundary conditions are enforced before starting the Runge-Kutta procedure. This is especially important for the multigrid process, because the coarse grid solution is injected from the fine grid and may have little in common with the previous coarse grid solution.

The second step involves the Runge-Kutta procedure itself. During this phase, the boundary conditions are implemented in two passes, as was shown in fig. 3. In the first pass, only cut boundaries, whose source data comes from higher blocks, are implemented. In the second pass, all non-cut boundaries plus cut boundaries whose source data come from the current or lower blocks are implemented. The effect of this procedure is illustrated in figure 5 for a simple two block topology that connects on one side, and for one stage of the Runge-Kutta time step. At the beginning of a Runge-Kutta stage, the data at all points are at the k-th level except the cut ghost points of block 1, which are at the k-1 level. After applying the first pass of boundary conditions all points in block 1 are at the k-th level. The Runge-Kutta stage advances the interior solution to the k+1 level, and the second pass of boundary conditions brings all ghost points of block 1, except the cut points, also to the k+1 level. Moving to block 2, the first pass has no effect because the source data for this cut lies in a lower block. The Runge-Kutta stage advances the interior points to the k+1 level, and the second pass of boundary conditions brings all ghost points to the same level. At the completion of the block loop, all points are at the initial level plus one. More importantly, at the start of each Runge-Kutta stage, all ghost points are at the same level as the interior points.

This two pass procedure works well for simply connected domains, but it does not entirely eliminate the lag for more complex topologies. This brings us to the third step. Figure 6 illustrates a topology with three blocks which overlap at a corner (in 3-D this represents a line of data). The corner ghost point of block 3 maps to a ghost point of block 2 which in turn maps to an interior point in block 1. The indirect mapping occurs whenever the corner point is treated as a contiguous part of two overlapping sides, and results in a lag after the two pass procedure. In the most complex case of 8 blocks all intersecting on a single corner, the number of mappings required to connect the ghost point with the appropriate interior point is greater. The corner point can be brought to the correct level by cycling through the cut boundaries a sufficient number of times; however, the I/O cost is prohibitive. The obvious remedy is to directly map the corner point from the correct interior point, rather than try to connect it contiguously with one of the sides. Within the present boundary data structure, this is easily done simply by treating the corner point at a separate cut.

Validation and Results

The validation consists of direct comparison of single block and multi-block results. Several different single block grids were split along coordinate planes to give two block grids which are geometrically identical to the single block case. All multi-block computations are performed with the same parameters (time stepping, smoothing, etc) as the associated single block case. No attempt is made to tune any of the parameters to optimize the result of any case. Comparing the single block base-line results with the multi-block computations gives a direct way to assess the accuracy and efficiency of the multi-block algorithm. A thorough validation of the basic flow solver for single block domains is presented in reference 1.

Computations were performed for viscous and inviscid flows about an ONERA-M6 wing. Both cases are at a Mach number of 0.84 and an angle of attack of 3.06 °. The Reynolds number for the viscous case is 6 million based on the mean chord. Also presented are inviscid solutions for the flow over the DLR-F4 wing-body combination at a Mach number of .75 and angle of attack of .84°.

The first set of calculations were performed out-of-core on a Cray-XMP with 16MW of core memory and 64MW SSD memory. The small core size limited the size of the test cases to under 250 thousand points ($\approx 62^3$ points). The inviscid case was computed on an O-O grid topology with 128x32x32 cells. The viscous case employed a C-O grid topology with 112x24x16 cells (16 spanwise). The inviscid grid was split along each coordinate direction to give three multi-block cases. The viscous grid was split only in the i and k directions. The present implementation of the Baldwin-Lomax turbulence model requires any turbulent region to reside in the same block as the wall from which the turbulent distance function is measured. The present grid was too coarse to allow a splitting in the j-direction.

Baseline results for the inviscid case are shown in figures 7 a-e. Comparison of the three multi-block case with the base-line is shown in figures 8 a-d. The convergence histories are essentially identical to the base-line for all three splittings, as are the Cp distributions at the wing root section. Although the i-split case places the cut at the leading edge, there is no visible difference there. The cut boundaries for the other cases do not intersect the wing root and therefore the good agreement is to be expected. At the spanwise station of the cut, the k-split case gives a slightly thicker shock than the base-line. The Cp contours on the wing upper surface show that the cut of the k-split case is close to the bifurcation point in the wing shock structure. The deviation is attributed to the approximations made to the smoothing term at cut boundaries. A similar set of results is presented for the viscous case: figures 9 a-e, show the base-line results, figures 10 a-d compare the multi-block cases with the baseline. As with the inviscid case, the convergence histories and Cp distributions show little deviation from the base line case, except for the k-split case which also thickens the shock.

Calculations for the last case, the DLR-F4 wing-body, were performed in-core on a Cray-2. The surface grid (coarser than actually used) is shown in figure 11. The topology is similar to a H-O topology except the entire k-plane maps onto the fuselage. The single block grid, having 208x24x64 cells, was split in the k-direction to form upper and lower blocks. Figure 12 a-d compare the convergence histories and Cp distributions on the wing. The Cp distribution on the fuselage is given in figure 13. There are no significant differences between the single block case and the multi-block case.

The computational overhead, measured in terms of CPU time, varies from 8% to 20% depending on the case. Much of the overhead can be attributed to two sources. When a single block domain is split, the cut plane is duplicated in the second block increasing the total number of points. This increase was as much as 14% in the case of the viscous grid split in the k-direction. The second source of overhead is the reduction of loop length that accompanies the decrease in block size. Where possible, the major routines have been coded to loop over the entire block in one loop so as to minimize this effect. In figure 14 a and b, the computational rate (in thousands of points per second) is plotted versus the block size (in thousands of points). The figure shows the multi-block case along with a single block (non-multigrid) case at different grid sizes. Note that the single block case experiences a large reduction in computational rate due to the reduction in the total number of points.

Conclusions

A multi-block algorithm has been developed and validated for inviscid and viscous flows about aircraft configurations. The method preserves the convergence and accuracy properties of the original flow solver. The method is not sensitive to the introduction of new cut planes at block boundaries. This allows domains to be blocked primarily on geometric considerations rather than flow considerations. Computational overhead varies from 8% to 20% depending on the case; however, much of the overhead is attributed to the decrease in block size that results from splitting a single block grid. In a realistic situation, in which each block is sufficiently large, the overhead is not expected to be significant.

References

1. Radespiel, R., "A Cell-Vertex multigrid method for the Navier-Stokes Equations," NASA TM 101557, January 1989.

2. Ames,W. F., *Numerical Methods for Partial Differential Equations,* Academic Press, New York, 1977.

3. Martinelli, L., "Calculations of Viscous Flows with a Multigrid Method," Ph.D. Dissertation, MAE Department, Princeton University, 1987.

4. Jameson, A., Schmidt, W., Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping-Schemes," AIAA Paper 81-1259, 1981.

5. Whitfield, D.L., Janus, J.M., "Three-Dimensional Unsteady Euler Equations Solutions Using Flux Vector Splitting," AIAA Paper 84-1552, 1984.

6. Rossow, C., Kroll, N., Radespiel, R., Scherr, S., "Investigation of the Accuracy of Finite Volume Methods For 2- and 3-Dimensional Flows," AGARD Conference: Validation of Computational Fluid Dynamics, Vol.1, 1989.

7. Kroll, N., Rossow, C., Scherr, S., Schone, J., Wichmann, G., "Analysis of 3-D Aerospace Configurations Using the Euler Equations," AIAA paper 89-0268, 1989.

8. Fritz, W., "Numerical Simulations of 2D Turbulent Flow Fields with Strong Separation," ICAS-88-4.6.4, 16th Congress of the ICAS, Jerusalem, 1988.

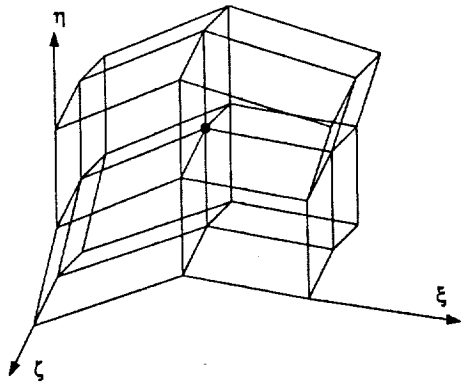


Fig. 1 Structure of a super-cell about a grid point

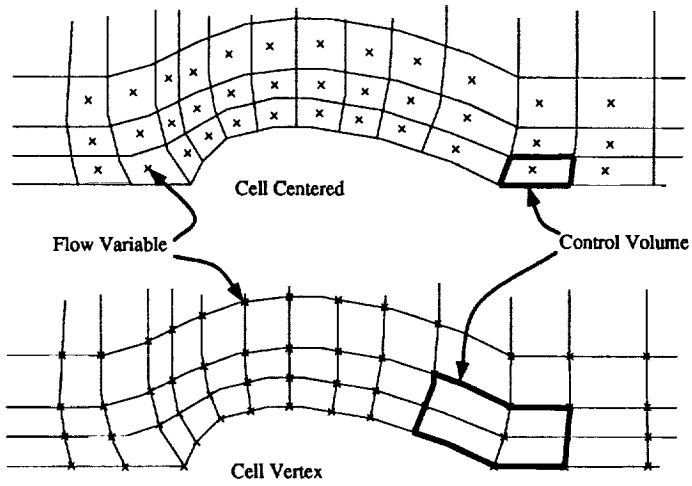


Fig. 2 A comparison of cell centered and cell vertex methods

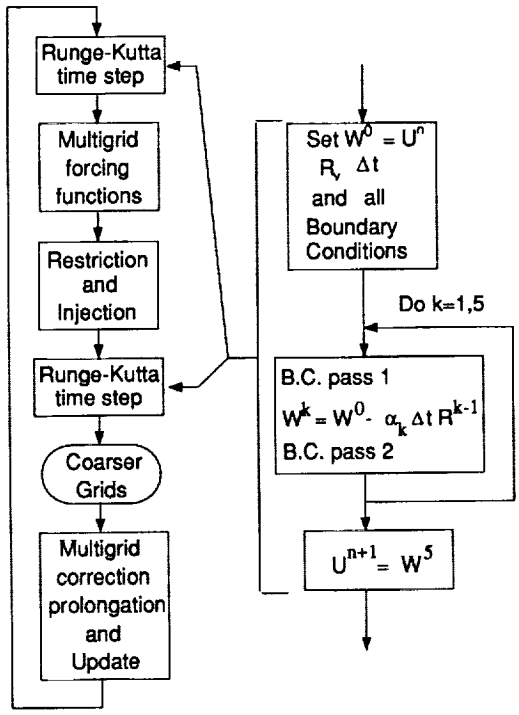


Fig. 3 Block diagram of the multi-block algorithm

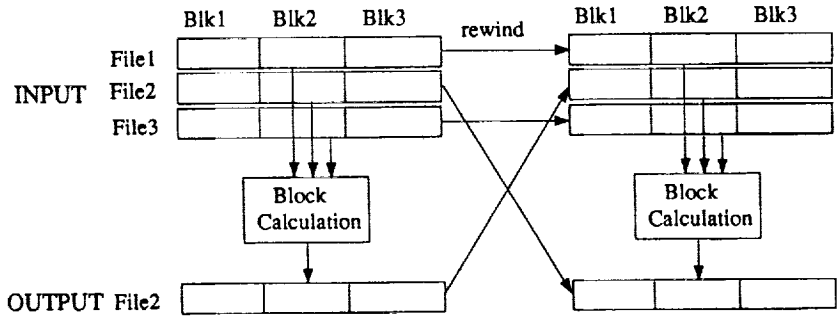


Fig. 4 File structure and I/O flow for out-of-core calculation

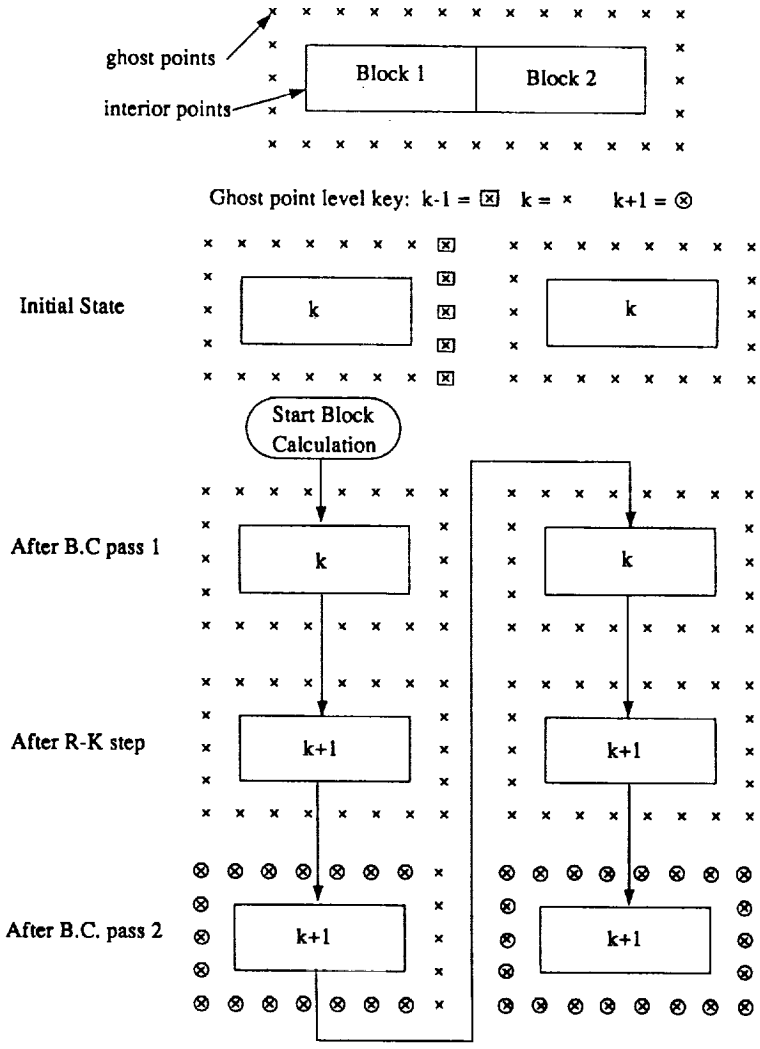


Fig. 5 Block diagram of the implementation of boundary conditions within the Runge-Kutta time step

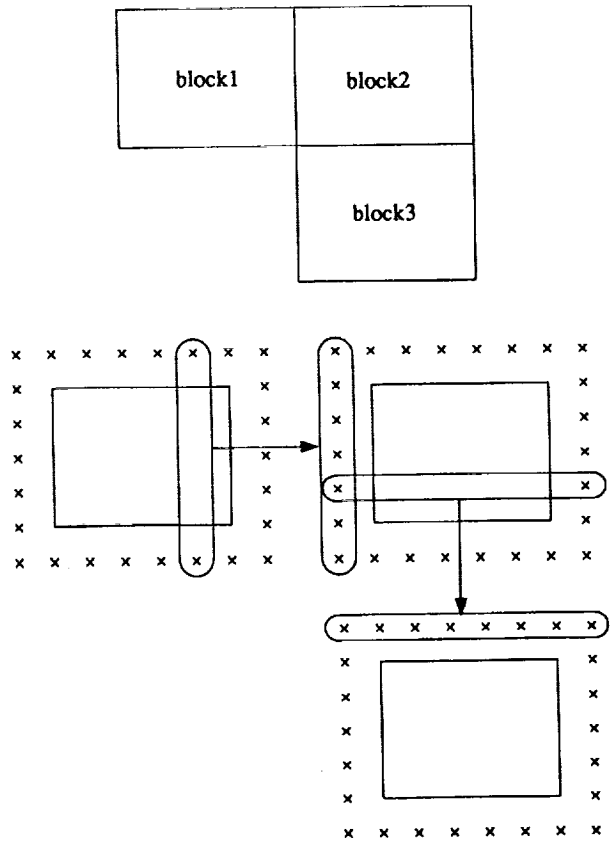
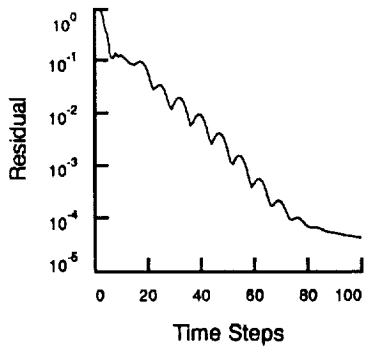
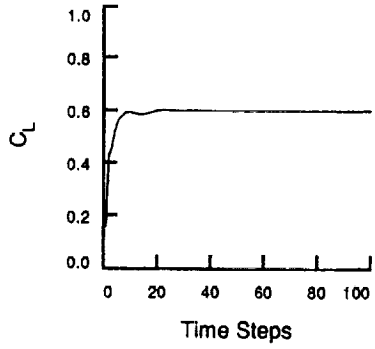


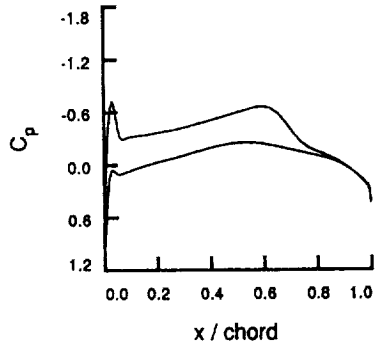
Fig. 6 A three block topology for which the two pass boundary procedure does not eliminate all lag



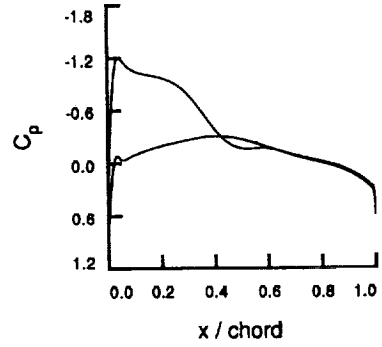
a) Residual convergence



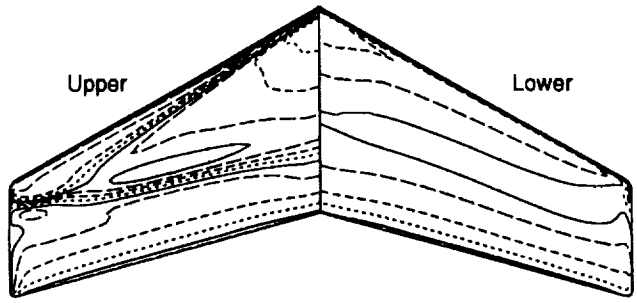
b) Convergence of lift coefficient



c) Pressure distribution at η=0.0

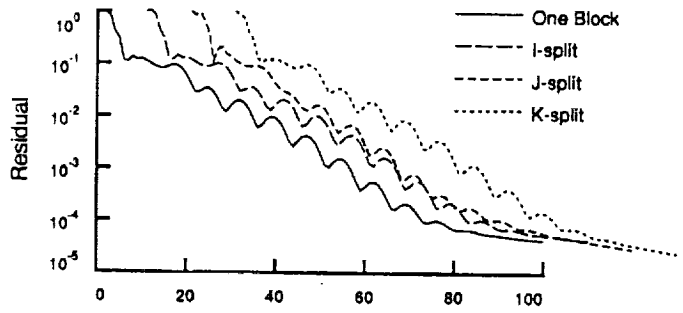


d) Pressure distribution at η=0.79

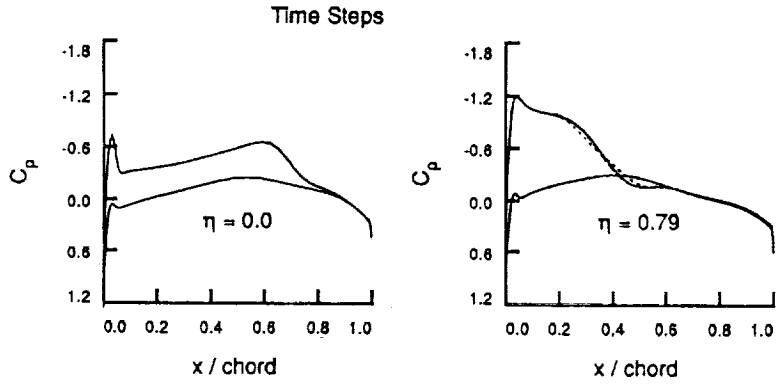


e) Pressure contours on wing surface

Fig. 7 Single block Euler solution for flow over an ONERA-M6 wing at M = 84, α = 3.06

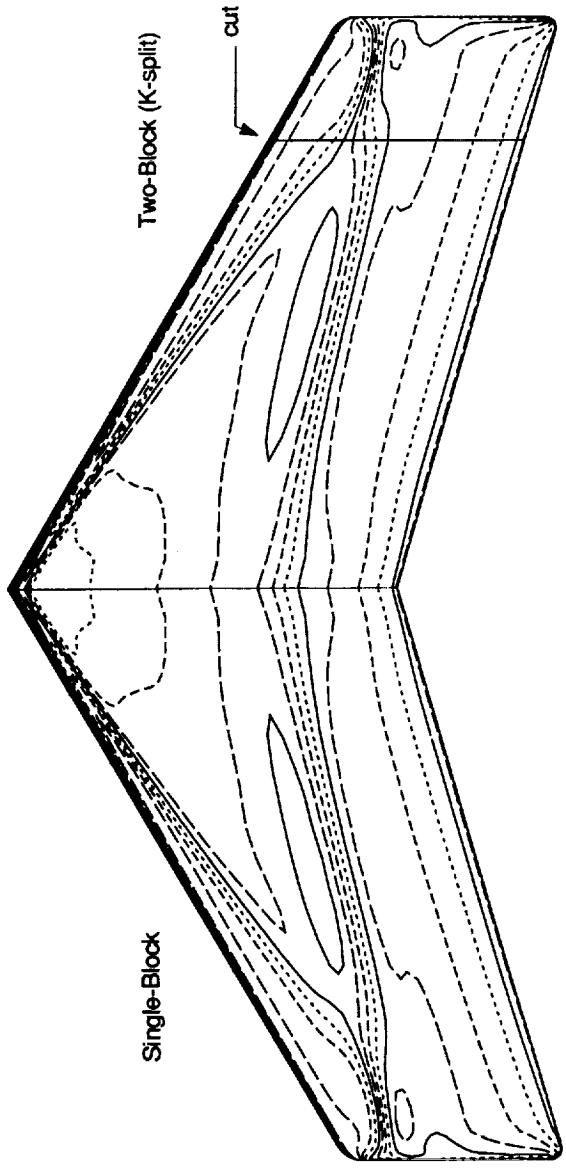


a) Influence of block structure on residual convergence



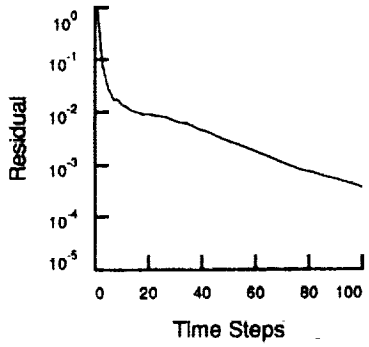
b) Influence of block structure on pressure distributions

Fig. 8 Comparison of single and multi-block inviscid solutions

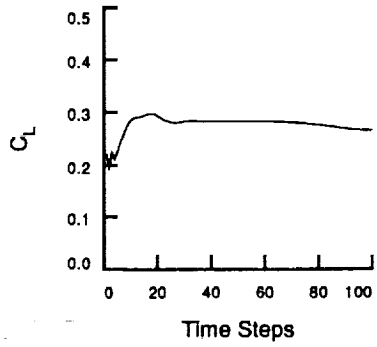


c) Influence of block structure on pressure contours

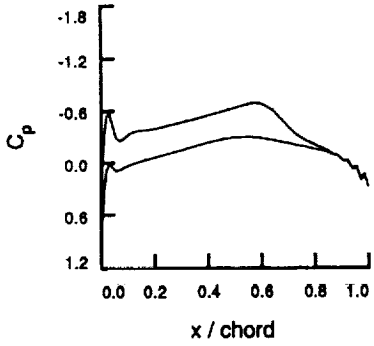
Fig. 8 continue



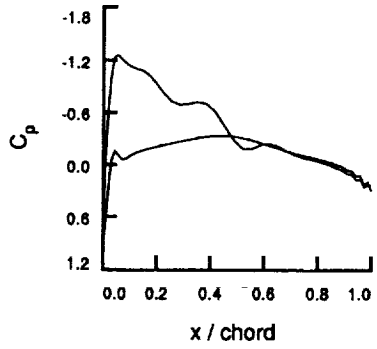
a) Residual convergence



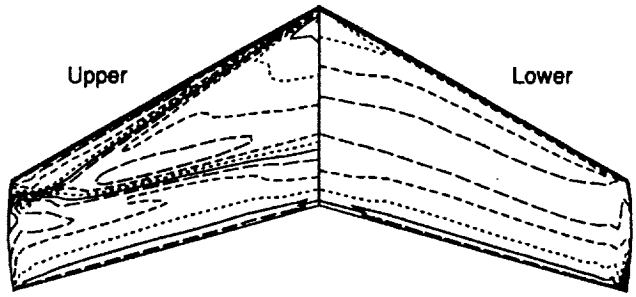
b) Convergence of lift coefficient



c) Pressure distribution at η=0.0

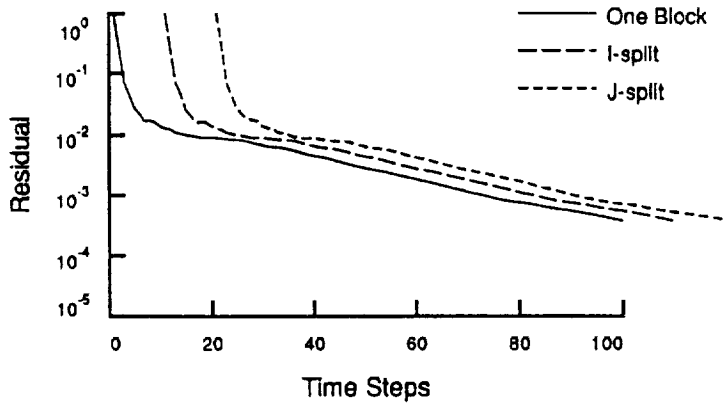


d) Pressure dsitribution at η=0.79

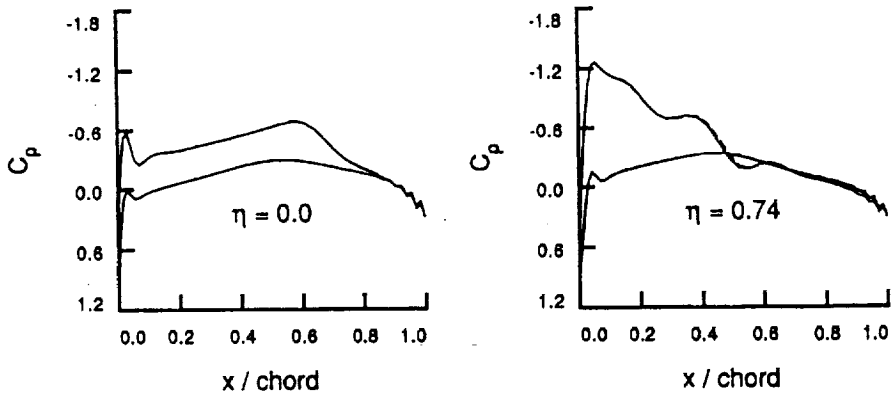


e) Pressure contours on wing surface

Fig. 9 Single block Navier-Stokes solution for flow over an ONERA-M6 wing at M = 84, α = 3.06, Re = 11. x 10^{6}

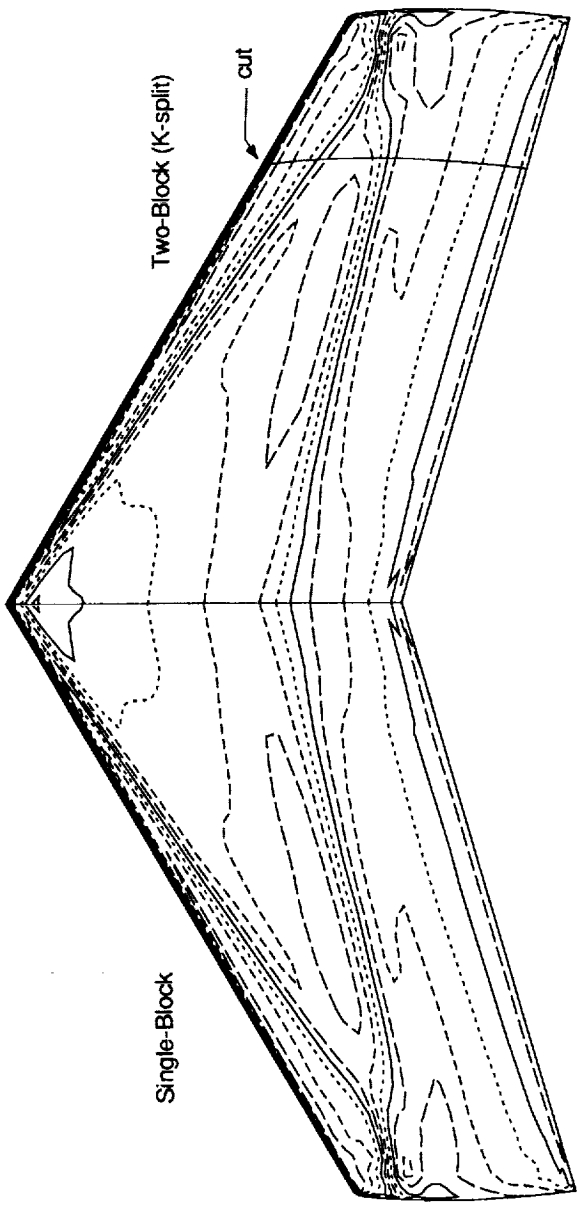


a) Influence of block structure on residual convergence



b) Influence of block structure on pressure distributions

Fig. 10 Comparison of single and multi-block solutions



c) Influence of block structure on pressure contours

Fig. 10 continue

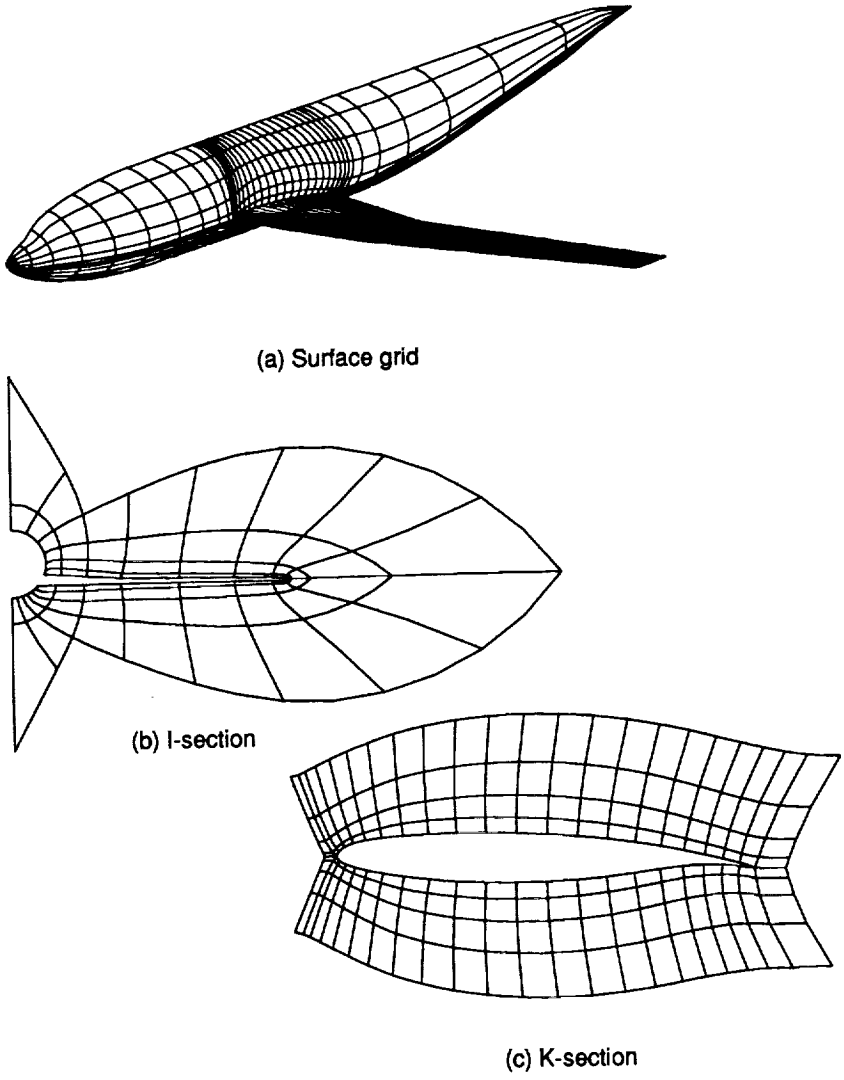
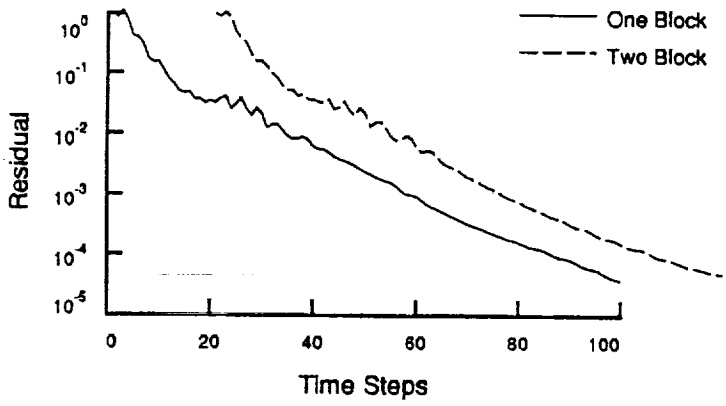
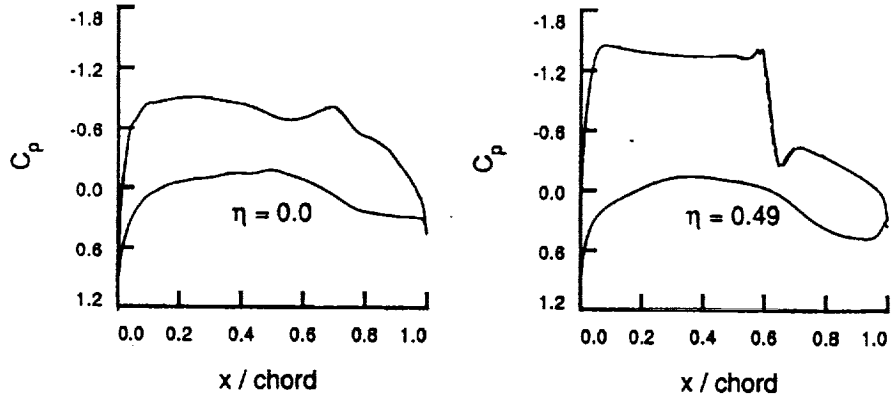


Fig. 11 Surface grid, i=constant plane and j=constant plane for the grid about the DLR-F4 wing-fuselage

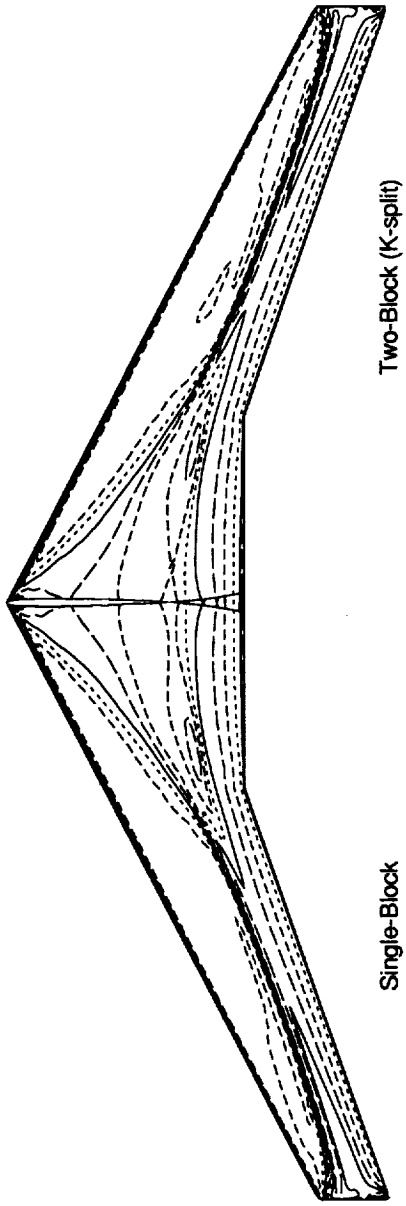


a) Influence of block structure on residual convergence



b) Influence of block structure on pressure distributions

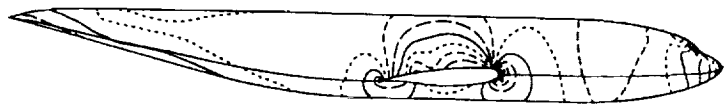
Fig. 12 Comparison of single and multi-block solutions for inviscid flow over the DLR-F4 wing-fuselage at $M = 0.75$, $\alpha = 0.84$



c) Influence of block structure on pressure contours

Fig. 12 continue

One Block



Two Blocks

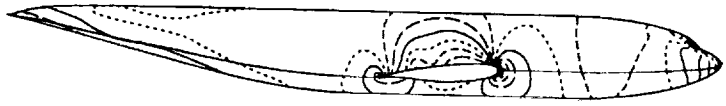
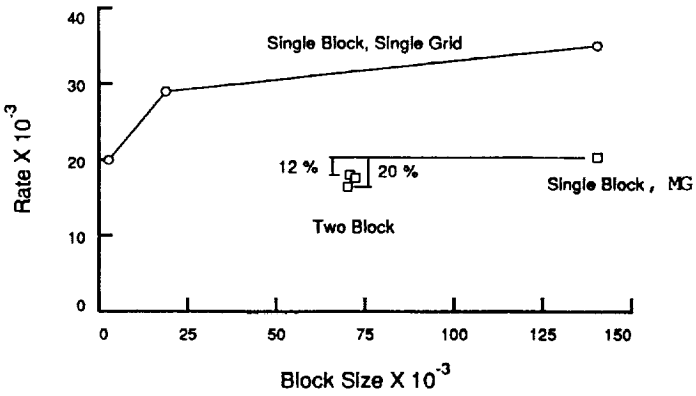
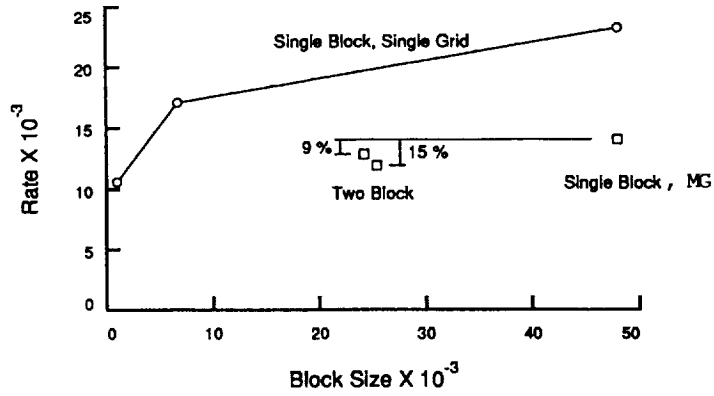


Fig. 13 Comparison of C_p contours on the fuselage for single and multi-block solutions.



a) Influence of block size and block structure on computation rates of the Euler version of the code



b) Influence of block size and block structure on computation rates of the Navier-Stokes version of the code

Fig. 14 The computational rate Vs. block size for single- and multi-block computations

Forschungsbereich Flugmechanik/Flugführung
Bereichsleitung: Flughafen, D-3300 Braunschweig

- Institut für Flugmechanik
- Institut für Flugführung
- Institut für Dynamik der Flugsysteme
- Institut für Flugmedizin
- Hauptabteilung Verkehrsforschung

Forschungsbereich Strömungsmechanik
Bereichsleitung: Bunsenstraße 10, D-3400 Göttingen

- Institut für Theoretische Strömungsmechanik
- Institut für Experimentelle Strömungsmechanik
- Institut für Antriebstechnik
- Institut für Entwurfsaerodynamik

Forschungsbereich Werkstoffe und Bauweisen
Bereichsleitung: Pfaffenwaldring 38–40, D-7000 Stuttgart 80

- Institut für Strukturmechanik
- Institut für Aeroelastik
- Institut für Werkstoff-Forschung
- Institut für Raumsimulation
- Institut für Bauweisen- und Konstruktionsforschung

Forschungsbereich Nachrichtentechnik und Erkundung
Bereichsleitung: Oberpfaffenhofen, D-8031 Weßling/Obb.

- Institut für Nachrichtentechnik
- Institut für Hochfrequenztechnik
- Institut für Optoelektronik
- Institut für Physik der Atmosphäre

Forschungsbereich Energetik
Bereichsleitung: Pfaffenwaldring 38–40, D-7000 Stuttgart 80

- Institut für Technische Physik
- Institut für Technische Thermodynamik
- Institut für Physikalische Chemie der Verbrennung
- Institut für Chemische Antriebe und Verfahrenstechnik

Bereich Wissenschaftlich-Technische Betriebseinrichtungen
Bereichsleitung: Oberpfaffenhofen, D-8031 Weßling/Obb.

Projektträgerschaft Weltraumforschung/Weltraumtechnik
Leitung: Linder Höhe, D-5000 Köln 90 (Porz)

Managementdienste

Projektträgerschaften für Arbeit, Umwelt und Gesundheit
Leitung: Südstraße 125, D-5300 Bonn 2

Veröffentlichungen der DLR

DLR-Forschungsberichte
Wissenschaftliche Erstveröffentlichungen aus der Forschungs- und Entwicklungsarbeit der DLR.

DLR-Mitteilungen
Beiträge über Versuchsmethoden und -anlagen, Rechenprogramme, Vortragsveranstaltungen, Literaturübersichten zu bestimmten Themenkreisen.

Jahresverzeichnis
der DLR-Forschungsberichte und DLR-Mitteilungen. (deutsch und englisch)

Zeitschrift für Flugwissenschaften und Weltraumforschung (ZFW)
Wissenschaftliche Zeitschrift mit Beiträgen aus der Luft- und Raumfahrtforschung und -technologie.

DLR-Nachrichten
Zeitschrift mit Beiträgen über aktuelle Forschungsarbeiten und mit Hinweisen auf DLR-Veröffentlichungen, DLR-Erfindungen und -Patente, Projektträgerberichte und Veranstaltungen.

Wissenschaftliche Berichte der Bereiche
Übersicht über die wesentlichsten Forschungs- und Entwicklungsarbeiten, Struktur, Aufgabenspektrum der Bereiche sowie Ausblick auf die zukünftige Entwicklung. (deutsch und englisch)

Erfindungen und Patente
Liste mit erteilten bzw. angemeldeten Patenten.